

AD-A227 893

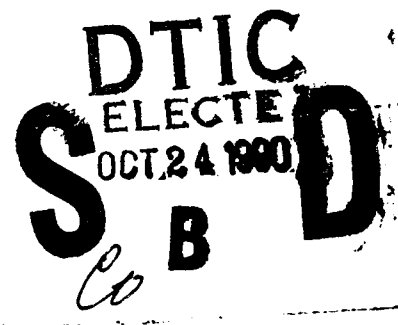
RADC-TR-90-195, Vol I (of two)
Final Technical Report
September 1990



THE ASSISTANT FOR SPECIFYING THE QUALITY SOFTWARE (ASQS) Operational Concept Document

Dynamics Research Corporation

Larry Kahn and Steve Keller



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

90 10 23 163

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-90-195, Vol I (of two) has been reviewed and is approved for publication.

APPROVED:



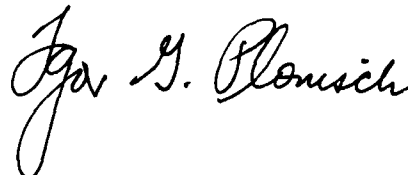
ROGER J. DZIEGIEL, JR.
Project Engineer

APPROVED:



RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command and Control

FOR THE COMMANDER:



IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COEE) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Project, Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE		3. REPORT TYPE AND DATES COVERED	
				Final Mar 88 - Jan 90	
4. TITLE AND SUBTITLE				5. FUNDING NUMBERS	
THE ASSISTANT FOR SPECIFYING THE QUALITY SOFTWARE (ASQS) Operational Concept Document				C - F30602-87-D-0086 PE - 63728F PR - 2527 TA - Q4 WU - 03	
6. AUTHOR(S)					
Larry Kahn, Steve Keller					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)				8. PERFORMING ORGANIZATION REPORT NUMBER	
Dynamics Research Corporation 60 Frontage Road Andover MA 01810					
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
Rome Air Development Center (COEE) Griffiss AFB NY 13441-5700				RADC-TR-90-195, Vol I (of two)	
11. SUPPLEMENTARY NOTES					
RADC Project Engineer: Roger J. Dziegiel, Jr./COEE/(315) 330-4063					
12a. DISTRIBUTION/AVAILABILITY STATEMENT				12b. DISTRIBUTION CODE	
Approved for public release; distribution unlimited.					
13. ABSTRACT (Maximum 200 words)					
<p>The ASQS, a knowledge base software tool, assists software acquisition managers which specify software quality requirements. In the consultation mode, questions are presented to the acquisition manager in terms related to the mission area of interest. Answers are transformed by ASQS into tailored quantitative quality goals based on the thirteen software quality factors of the RADC software quality framework. The current version supports the Intelligence and Satellite mission areas.</p> <p>ASQS serves the purpose of transitioning the software quality specification methodology found in the RADC final report RADC-TR-85-37, Vol II (of three), "Specification of Software Quality Attributes - Software Quality Specification Guidebook", into use in the DOD acquisition process. In so doing, it reduces the amount of time and the expertise required to specify meaningful software quality goals. It bridges the gap between software quality concepts and terminology and the system needs and terminology understood by the acquisition manager. <i>Keywords:</i></p> <p style="text-align: right;">(continued)</p>					
14. SUBJECT TERMS				15. NUMBER OF PAGES	
Software, Quality, Specification, Knowledge base, Software Quality Metrics. <i>(KR)</i>				100	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT		18. SECURITY CLASSIFICATION OF THIS PAGE		19. SECURITY CLASSIFICATION OF ABSTRACT	
UNCLASSIFIED		UNCLASSIFIED		UNCLASSIFIED	
				20. LIMITATION OF ABSTRACT	
				SAR	

Block 13 (Continued)

ASQS was developed using DOD-STD-2167 dated June 1985. The tool is decomposed into five top level functions: 1) user interface to implement the window interface, 2) ranker which ranks the quality factors and criteria in priority order, 3) quantifier which generates numeric factor requirements based on an evaluation of the tailored framework, 4) assessor which assesses compliance of the measurements as compared to the required framework, and 5) administrator which handles all administrative functions such as user access and security. The ASQS has been designed to interface with the Quality Evaluation System (QUES) which evaluates the quality of a software system. A tailored framework with quality goals is transported to QUES from the quantifier function. Measurements by QUES are then transported to the assessor function to assess compliance with goals after each life cycle phase.

Features have been implemented to support tool usability. Windows, menus and use of a mouse help make ASQS user friendly. Generic systems for each mission area have been decomposed into functions. For the decompositions of the Intelligence and Satellite areas, rule sets have been developed and tailored. These can be used as references and copied in part to a new system being developed. Other features which have been incorporated include capturing the rationale behind all elements of the quality specification, recording the history of all changes, identifying the location of problems during assessment, and allowing the generation of what if scenarios by changing answers to questions and functional decomposition. Consideration of software quality requirements is now feasible as early as concept exploration and changes to the original and subsequent specifications can be recorded through post deployment.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



TABLE OF CONTENTS

<u>SECTION</u>		<u>PAGE</u>
1.	SCOPE	1
1.1	IDENTIFICATION	1
1.2	PURPOSE	1
1.3	INTRODUCTION	1
2.	REFERENCED DOCUMENTS	3
3.	MISSION	4
3.1	MISSION NEED REQUIREMENTS	4
3.2	PRIMARY MISSIONS	7
3.3	SECONDARY MISSIONS	9
3.4	OPERATIONAL ENVIRONMENT	11
3.4.1	USER PROFILE	11
3.4.2	RELATIONSHIP BETWEEN DOD-STD-2167 AND THE ASSISTANT	15
3.4.3	RELATIONSHIP BETWEEN PROPOSED DOD-STD-2168 AND THE ASSISTANT	15
3.5	SUPPORT ENVIRONMENT	16
3.5.1	HARDWARE SUPPORT	16
3.5.2	SOFTWARE SUPPORT	16
3.5.3	SOFTWARE REQUIRED	16
3.5.4	EQUIPMENT REQUIRED	16
3.5.5	FACILITIES REQUIRED	17
3.5.6	SYSTEM ADMINISTRATOR REQUIREMENTS	17
4.	SYSTEM FUNCTIONS AND CHARACTERISTICS	18
4.1	SYSTEM FUNCTIONS	18
4.1.1	RANK FACTORS AND CRITERIA	18
4.1.1.1	IDENTIFY APPLICATION/FUNCTIONS	19
4.1.1.2	ASSIGN QUALITY FACTORS AND RANKINGS	20
4.1.1.2.1	APPLICATION CHARACTERISTICS	21
4.1.1.2.2	ENVIRONMENT CHARACTERISTICS	21
4.1.1.2.3	DEVELOPMENT CHARACTERISTICS	22
4.1.1.2.4	SYSTEM QUALITY FACTORS	22
4.1.1.2.5	SURVEY RESULTS	23
4.1.1.2.6	SPECIFIC QUALITY CONCERNS	23
4.1.1.2.7	COMPLEMENTARY FACTORS	23
4.1.1.2.8	COMBINING SELECTIONS TO DETERMINE RANKINGS	24
4.1.1.3	CONSIDER INTERRELATIONSHIPS	24
4.1.1.3.1	SHARED CRITERIA	25
4.1.1.3.2	POSITIVE AND NEGATIVE FACTOR INTERRELATIONSHIPS	25
4.1.1.4	CONSIDER COSTS	26
4.1.2	QUANTIFY FACTORS	26
4.1.2.1	ACQUISITION QUESTIONS	27
4.1.2.2	BASELINE VALUES	30
4.1.2.3	DATA ITEM COUNTS	31
4.1.2.4	CALCULATIONS	31
4.1.3	ASSESS COMPLIANCE	32
4.1.4	SYSTEM ADMINISTRATION FUNCTIONS	33

TABLE OF CONTENTS (CONTINUED)

<u>SECTION</u>		<u>PAGE</u>
4.2	COMPUTER SYSTEM FUNCTIONS	34
4.2.1	REPORTS	34
4.2.1.1	SNAPSHOT REPORT	34
4.2.1.2	GOALS AND RATIONALE REPORT	34
4.2.1.3	QUESTIONS AND RATIONALE REPORT	34
4.2.1.4	TAILORED FRAMEWORK REPORT	35
4.2.1.5	CHANGE REPORT	35
4.2.1.6	ASSESSMENT OF COMPLIANCE REPORT	35
4.2.1.7	PERCENT UNKNOWN REPORT	36
4.2.2	REASONING	36
4.2.3	EXPLANATIONS	37
4.2.4	KNOWLEDGE ACQUISITION	38
4.2.5	WHAT-IF ANALYSIS	38
4.3	USER INTERACTION	38
4.4	COMPUTER SYSTEM CHARACTERISTICS	77
4.4.1	HARDWARE	77
4.4.2	SOFTWARE	77
5.	GLOSSARY	79
APPENDIX A	SAMPLE RULES FOR TAILORING PORTABILITY	85

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
3.1-1	THE SOFTWARE QUALITY SPECIFICATION METHODOLOGY ALLOWS ACQUISITION MANAGERS TO CONTROL SOFTWARE QUALITY	6
3.3-1	THE ASSISTANT PROVIDES A MECHANISM FOR CAPTURING KNOWLEDGE ABOUT SOFTWARE QUALITY	10
3.4-1	SPECIFICATION OF SOFTWARE QUALITY DURING THE CONCEPT EXPLORATION PHASE	13
3.4-2	SPECIFICATION OF SOFTWARE QUALITY DURING THE DEMONSTRATION AND VALIDATION PHASE	14
4.1-1	INFERENCES BASED ON ANSWERS TO QUESTIONS	29
4.3-1	THE ASSISTANT EXECUTES IN A SYSTEM SOFTWARE ENVIRONMENT	40
4.3-2	A TOP-LEVEL ROADMAP ILLUSTRATES THE SPECIFICATION STEPS	41
4.3-3	A SECOND-LEVEL ROADMAP ILLUSTRATES THE STEPS OF RANK FACTORS AND CRITERIA	42
4.3-4	MENUS GUIDE USERS THROUGH THE SUB-STEPS WITHIN A STEP ON THE ROADMAP	43
4.3-5	SELECTION OF MISSION AREA AND SOFTWARE TYPE ALLOWS TAILORING TO USER NEEDS	44
4.3-6	EXAMPLES PROVIDE FOR DECISION-MAKING BASED ON EXPERIENCE FROM OTHER PROGRAMS	45
4.3-7	SPECIFYING A NEW SYSTEM BASED ON PAST EXPERIENCE	46
4.3-8	A USER CAN MOVE THROUGH THE HIERARCHY OF SYSTEM AND SOFTWARE FUNCTIONS	47
4.3-9	A USER CAN TAILOR THE SYSTEM DECOMPOSITION TO THEIR SYSTEM	48
4.3-10	HISTORY PROVIDES THE BASIS FOR INFORMED DECISION-MAKING IN THE FUTURE	49
4.3-11	THE VERSION TREE PROVIDES ACCESS TO HISTORY	50
4.3-12	ASSIGN FACTORS AND CRITERIA RANKINGS	51
4.3-13	OPTIONAL PATHS ALLOW USERS TO SPECIFY AS THEY CHOOSE	52
4.3-14	THE SPECIFICATION IS REVIEWED TO TAILOR IT TO THE STARS SEE	53
4.3-15	THE USER ZOOMS IN TO REVIEW A FACTOR RANKING	54
4.3-16	REVIEWING SUGGESTINGS BY THE ASSISTANT BUILDS CONFIDENCE IN THE SPECIFICATION	55
4.3-17	REVIEWING THE DETAILED REASONING CHAIN ALLOWS USERS TO DETERMINE REFINEMENTS	56
4.3-18	QUALITY CONCERNS MAY BE SELECTED FROM A VARIETY OF SOURCES	57

LIST OF FIGURES (CONTINUED)

<u>FIGURE</u>		<u>PAGE</u>
4.3-19	SYSTEM FACTOR CHARACTERISTICS ARE REVIEWED FOR APPLICABILITY	58
4.3-20	APPLICATION CHARACTERISTICS ARE QUICKLY REFINED BY TWO STEPS	59
4.3-21	LISTS OF CONCERNS HELP ENSURE THAT ALL ISSUES ARE CONSIDERED BY THE USER	60
4.3-22	THE IMPLICATIONS OF CHANGES TO ANSWERS ARE DETERMINED DURING A CONSULTATION	61
4.3-23	IMPLICATIONS ARE SUMMARIZED IN THE FACTOR RANKING WINDOW	62
4.3-24	THE RANKINGS ARE REVISED BASED ON FACTOR INTERRELATIONSHIPS	63
4.3-25	NOTES DRAW THE USER'S ATTENTION TO RECOMMENDED CHANGES BASED ON FACTOR INTERRELATIONSHIPS	64
4.3-26	THE USER REVIEWS REASONS FOR THE RECOMMENDED CHANGE	65
4.3-27	EACH REASON CAN BE REVIEWED FOR APPLICABILITY	66
4.3-28	THE EXPLANATION CAN BE EXPANDED INTO A COMPLETE RATIONALE	67
4.3-29	REVISE RANKINGS BASED ON COSTS	68
4.3-30	A SECOND-LEVEL ROADMAP ILLUSTRATES THE STEPS OF QUANTIFY FACTORS	69
4.3-31	ACQUISITION QUESTIONS ARE ANSWERED TO TAILOR THE FRAMEWORK	70
4.3-32	THE USER CAN ASK THE ASSISTANT FOR EXPLANATIONS OF THE QUESTIONS	71
4.3-33	A WHY EXPLANATION IS SHOWN	72
4.3-34	A TAILORED FRAMEWORK CAN BE REVIEWED	73
4.3-35	THE TAILORED FRAMEWORK IS REVIEWED BY PHASE	74
4.3-36	THE TAILORED FRAMEWORK IS SHOWN IN HIERARCHICAL FORM	75
4.3-37	NUMERICAL FACTOR GOALS ARE PROVIDED AUTOMATICALLY BASED ON THE FRAMEWORK AND USER INTERACTION	76
4.4-1	SYSTEM ASSISTANT'S HARDWARE CONFIGURATION	78

1. SCOPE

The identification, purpose and introduction are presented in the following sub-sections.

1.1 Identification

This Operational Concept Document describes the mission of the Assistant for Specifying the Quality of Software (ASQS -- pronounced "asks") and its operational and support environments. It also describes the functions and characteristics of the computer system within the overall system. The Assistant for Specifying the Quality of Software, which will be referred to throughout this document as the Assistant, is sponsored by the Rome Air Development Center (RADC/COEE) under contract number F30602-86-C-0157.

1.2 Purpose

The Assistant serves the purpose of transitioning the Specification Of Software Quality Attributes Guidebooks (Vol. I-III) [SPECSQ] and associated quality methodology into use in the DoD acquisition process. By automating Volume II, the Assistant reduces the amount of time and expertise required to specify meaningful software quality goals. The Assistant bridges the gap between software quality concepts, terminology, system needs, and terminology understood by DoD acquisition managers.

1.3 Introduction

This document is intended to give readers an understanding of the capabilities of the Assistant and an understanding of how these capabilities allow acquisition managers to specify quality by engaging in a dialogue with the Assistant in which the manager supplies system-specific characteristics and needs and the Assistant fills in the software quality concepts and methods.

The mission needs, and primary and secondary missions are described in Sections 3, 3.1, 3.2 and 3.3. The essence of the mission is to make the software quality specification technology available to managers who are not experts in software quality technology. Section 3.4 (Operational Environment) provides a profile of the Assistant's users. Section 3.5 provides a description of the support environment.

The system functions that support the specification of software quality are described in Section 4.1. These functions are based on the Specification of Software Quality Guidebooks (Vol. I-III) [SPECSQ] and several enhancements developed by DRC in the Guidebook Validation tasks (Contract F19628-84-D-0016, Tasks 80025 and 80073). Several important computer system functions, which provide general capabilities for supporting a variety of the system functions, are described in Section 4.2.

The user interaction is described in Section 4.3. The user interaction is presented in the form of a hypothetical scenario, illustrated by snapshots of the monitor. The scenario is intended to clarify how the Assistant is used. It does not represent a specification of the user interface.

Computer system characteristics are described in Section 4.4. A glossary is provided in Section 5.

2. REFERENCED DOCUMENTS

[ADARM] Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, United States Department of Defense, Washington, D.C., January 1983.

[GUIDEVAL] Software Quality Guidebook Validation Documents, Contract No. F19628-84-D-0016, Task 25, CDRLS 104 - 107, Task 73, CDRLS 104 - 110.

[SEEOCD] Stars-SEE Operational Concept Document, prepared for Stars Joint Program Office, 02 October 1985.

[SOFTIND] Software Management Indicators, Air Force Systems Command Pamphlet 800-43, 31 January 1986, Software Quality Indicators, Air Force Systems Command Pamphlet 800-XX, 29 July 1986.

[SOFTSD] Defense System Software Development, DOD-STD-2167, 04 June 1985.

[SOFTQE] Software Quality Evaluation, DOD-STD-2168, DRAFT, 26 April 1985.

[SOFTST] Software Test Handbook, RADC-TR-84-53, Vol. II, March 1984.

[SPECSQ] Specification of Software Quality Attributes, RADC-TR-85-37, Vol I-III, February 1985.

[TEIRESIAS] Barr, A. and Feigenbaum, E. A., The Handbook of Artificial Intelligence, Vol II, page 87-101 HeurisTech Press, Stanford, Ca., 1982.

3 MISSION

This section describes the need for the Assistant and the primary and secondary missions. The section concludes with a description of the operational and support environments.

3.1 Mission Need Requirements

"Software is a critical and major component of DoD systems. The generation and continuous improvement of this software to meet changing DoD requirements has become a major factor in the fielding of systems needed to meet Service missions. All too often software suffers from low quality, long deployment times, and high development and support costs. The highly complex systems of the future are likely to make even greater demands." [SEEOCD]

The software produced for DoD systems must demonstrate a variety of software quality factors, such as reliability, maintainability, portability, and flexibility, as well as cost-effectiveness. Meeting these goals requires rigorous techniques for selecting appropriate quality factors, balancing quality levels and cost, specifying goals, and evaluating achieved quality.

RADC has developed a definition of software quality and an increased understanding of the relationships among the factors that make up software quality. The Specification of Software Quality Attributes Guidebooks (Vol. I-III) [SPECSQ] provide acquisition managers with a methodology to consider, balance, specify and evaluate software quality requirements. This Methodology involves identifying quality goals, considering interrelationships, considering costs, and selecting and specifying quality criteria. The quality goals provide the baseline against which to evaluate quality, and the target of convergence for the iterative process of evaluating and improving quality, as shown in Figure 3.1-1.

Figure 3.1-1 shows the methodology in two major parts: software quality specification and software quality evaluation. Specification is the responsibility of software acquisition managers and includes specifying software quality requirements and assessing compliance with those requirements. The specification guidebook (Volume II) provides procedural guidance. The results are documented in a Software Quality Specification Report and in the Software Requirements Specification. Evaluation is the responsibility of data collection and analysis personnel and includes applying software quality metrics to products of the development cycle, assessing product quality levels, and reporting results. The evaluation guidebook (Volume III) provides procedural guidance. The results are documented in a Software

Quality Evaluation Report. Section 4.0 of Volume I provides an overview of these processes.

Transitioning the Guidebooks into DoD systems development is an important part of realizing the benefits of the methodology. Unfortunately, software quality technology is a difficult technology to transform into use. Several factors contribute to this:

- o The Specification Guidebook requires a substantial time investment to learn and to apply. Applying the Specification Guidebook effectively requires knowledge of software quality concepts and methods (including factors, criteria, metrics, metric-elements, interrelationships, tailoring, weighting, etc.), in addition to knowledge of the mission area and system specifics. Individuals who are responsible for high-level management and quality issues often do not have time to familiarize themselves with the more technical aspects of quality. Also, significant experience applying the Specification Guidebook is required before it can be used effectively.
- o The data collection and evaluation process is labor intensive and costly.

The costliness of data collection and evaluation is addressed by the development of the Quality Evaluation System (QUES) to automate the collection and evaluation process. The other factors are addressed by the Assistant for Specifying the Quality of Software.

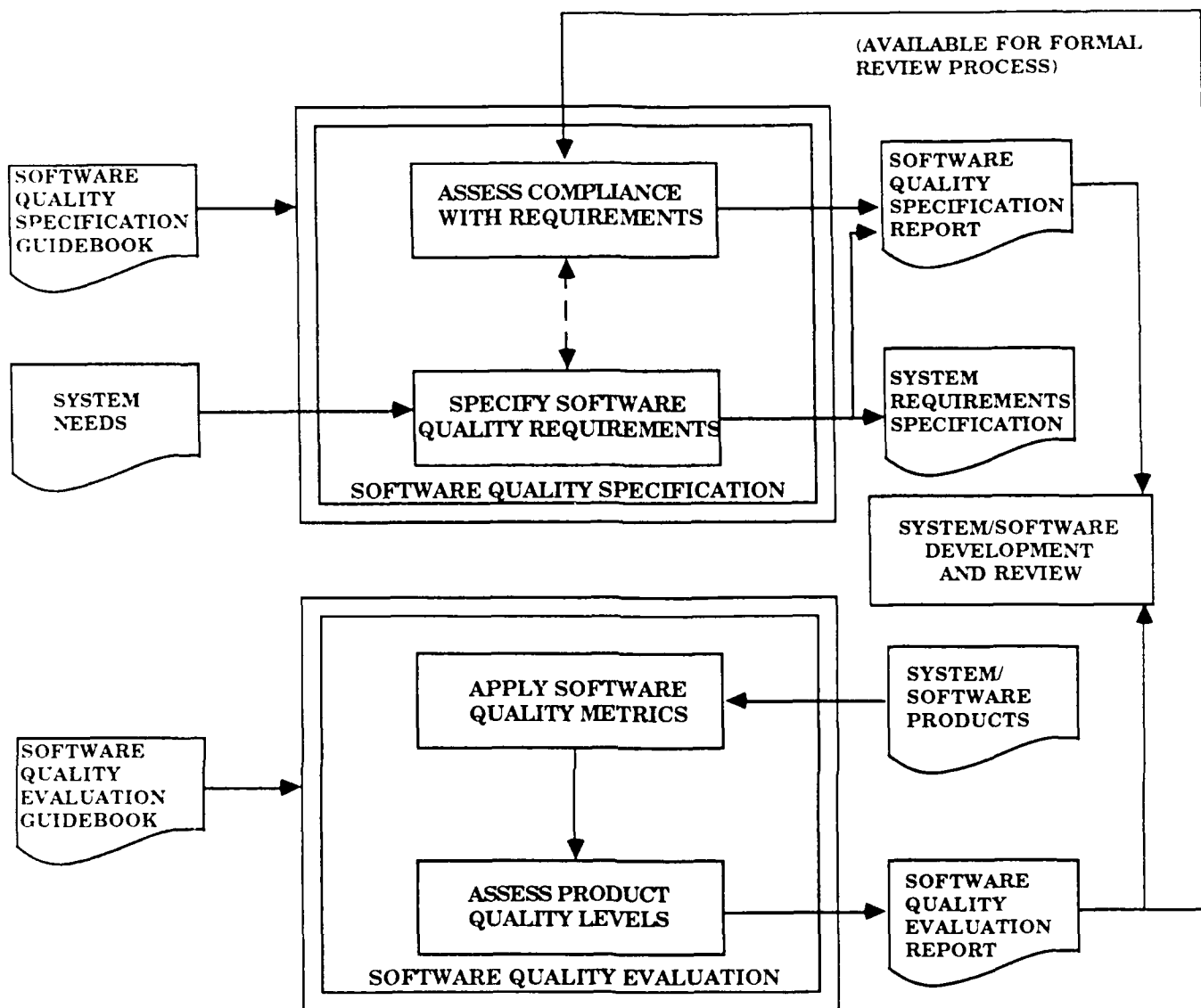


FIGURE 3.1-1 The Software Quality Specification Methodology Allows Acquisition Managers To Control Software Quality

3.2 Primary Missions

The primary mission of the Assistant is to make the Software Quality Specification Methodology accessible to DoD acquisition managers. The elements of this mission can be summarized as follows:

- o Assist acquisition managers in selecting appropriate software quality factors and in balancing quality levels and cost tradeoffs. Interact with the acquisition manager to translate his perception of the system characteristics and needs into required software quality factors.
- o Assist acquisition managers in selecting the required software characteristics needed to achieve the software quality factor goals. Interact with the acquisition manager to translate his perception of the system characteristics and needs into required software characteristics (criteria, metrics, metric-elements, and weightings).
- o Assist acquisition managers in determining the potential effect of different system characteristics and development approaches on software quality factor goals. For example, determine the effect of a particular DOD-STD-2167A tailoring on the expected quality.
- o Assist acquisition managers in assessing compliance of software with the specified quality goals.

The Assistant automates the software quality knowledge, specification methods, and the process of relating these to the mission area and system-specific knowledge. This allows users to specify software quality based on their knowledge of the mission area and system characteristics, without requiring them to develop and apply expert skills in software quality concepts and methods. The Assistant is an expert system in the sense that it provides the expertise of a software quality expert.

The Assistant allows users to develop a first-level quality specification in a short period of time (a few hours) and refine the specification throughout the program life as required. It permits specification prior to full-scale development, including concept exploration, and demonstration and validation.

The Assistant allows users to tailor the framework to their program without becoming embroiled in metric-level details. It plays the role of an expert who translates the user's high-level system characteristics and needs into low-level software characteristic requirements and presents the results in the user's high-level system requirements language.

Tailoring the framework to the characteristics of the program is essential because it allows the acquisition manager to obtain scores that are based only on applicable and relevant software characteristics. Without tailoring, the manager obtains scores that are biased by characteristics which are either not applicable (N/A), or not relevant to the system needs (i.e., they are added into the overall score). For example, the score for Application Independence may be biased too low because database management system independence is scored by the evaluators as a zero, when it should have been N/A because there is no database (this is an evaluation error that can occur when tailoring is not performed properly). Another possibility is that Application Independence is biased by database management system independence, when the database management system is intentionally specified by the acquisition team to be non-interchangeable.

If metric scores are biased by non-applicable or irrelevant characteristics, acquisition managers will lose confidence in the benefits of software quality metrics. Comparisons of metric scores between projects can be facilitated by providing the capability to add omitted metric scores into the overall scores upon request.

The capability to specify software quality requirements in terms of software characteristics is essential to the future goal of developing contractually binding software quality requirements. For most software quality factors, the observed quality rates (see Table 3.1-2. Quality Factor Ratings of the Guidebook), are dependent on two aspects: characteristics of the software itself, and characteristics of the environment in which the factor is observed. For example, maintainability rates are dependent on the effectiveness of the organization that maintains the software. Because of this consideration, contractual requirements for the maintainability rates are unlikely to be legally binding, particularly when the maintenance organization differs from the developer. However, requirements to meet specific software characteristics that have a direct impact on maintainability can clearly be levied on the development contractor. The Assistant allows users to specify those software characteristics based on acquisition concerns. The Assistant also provides the capability to graph relationships between software characteristics and observed quality rates once such data becomes available.

3.3 Secondary Missions

The secondary missions of the Assistant are, 1) to capture the history of selections, decisions and their rationale provided by software quality specifiers, and 2) provide an environment for viewing observed quality rates. The elements of these missions can be summarized as follows:

- o Capture selections, decisions, rationale, and the outcome of decisions to contribute to the rationale for the software quality specification at a given stage in the program life cycle.
- o Capture selections, decisions, rationale, and the outcome of decisions to contribute to specifications for future programs.
- o Capture selections, decisions, rationale, and the outcome of decisions to contribute to software quality research and enhancement of the specification methodology.
- o Allow users to view observed quality rates and their relationship to specified and achieved quality for programs with characteristics similar to their own.

The Assistant provides for the capture of domain knowledge about software quality concerns which spans the range of DoD applications and systems. The captured knowledge resulting from the application of the Assistant can be reviewed by software quality researchers for incorporation into the baseline specification methodology (see Figure 3.3-1).

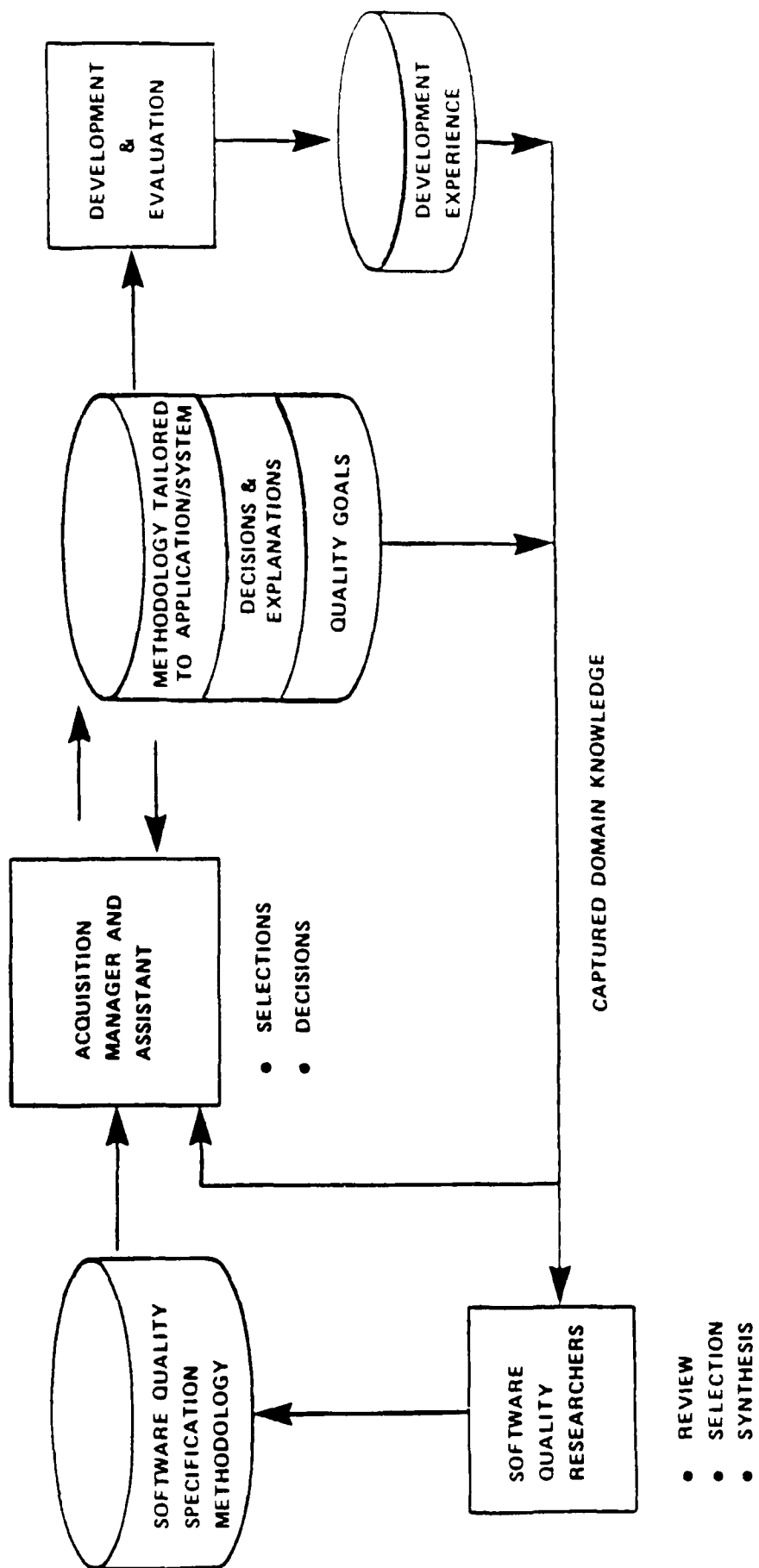


Figure 3.3-1. The Assistant Provides A Mechanism For Capturing Knowledge About Software Quality.

3.4 Operational Environment

The Assistant is a desk-top-based workstation used in an office environment. The system is used to help in project control and planning for software quality issues. Although the system is most effective for mission critical software, it can be an effective tool for specification of software quality for all types of systems. It will be shared by multiple users on multiple projects during all phases of the system acquisition life cycle (i.e., Concept Exploration, Demonstration and Validation, and Full Scale Development).

3.4.1 User Profile

The Assistant is flexibly designed to support a diverse user base. The system is used very early in the acquisition cycle (prior to Full Scale Development) by member(s) of the Computer Resources Working Group (CRWG) to aid in performing a software quality engineering study.

Figure 3.4-1 outlines the Assistant's role in performing specification of software quality engineering studies during the the Concept Exploration phase. System needs and quality concerns are used as inputs, resulting in preliminary software goals. This figure also shows how the software quality engineering study interacts with other ongoing engineering studies. For example, if one of the engineering studies is to determine system quality factors, this information would be used by the Assistant during the software quality engineering study (i.e., translation of system quality factors into software quality factors). The Assistant supports this type of interaction by utilizing information from a multitude of areas in the form of rules and accompanying rationale.

Figure 3.4-2 outlines the Assistant's role during the Demonstration and Validation phase. The results of earlier work, if available, are used as input. Refinements to the goals are made based on additional knowledge about the system. During this phase, engineering studies, prototypes, and additional planning occur. If a prototype is developed during this phase, the full range of the Assistant's capabilities are applicable.

After a SPO has been chosen, the project acquisition manager will use the Assistant to help identify system and software functions, identify important factors and criteria, specify factor rankings and goals, and evaluate the impact that tailoring of both DOD-STD-2167A and the Volume II worksheets has on quality.

Data collection and analysis personnel can use information such as system and software functions, quality goals, tailoring information, evaluation and criteria weighting formulas as input to the Quality Evaluation System, which will assist in the data collection and evaluation process.

Later in the life cycle, as actual evaluation of the software is performed, acquisition managers can use the results of evaluation as input to the Assistant to prepare an Assessment of Compliance Report, which documents details regarding the identification of quality deficiencies, tracking of quality trends, and, if necessary, updating of the initial factor goals.

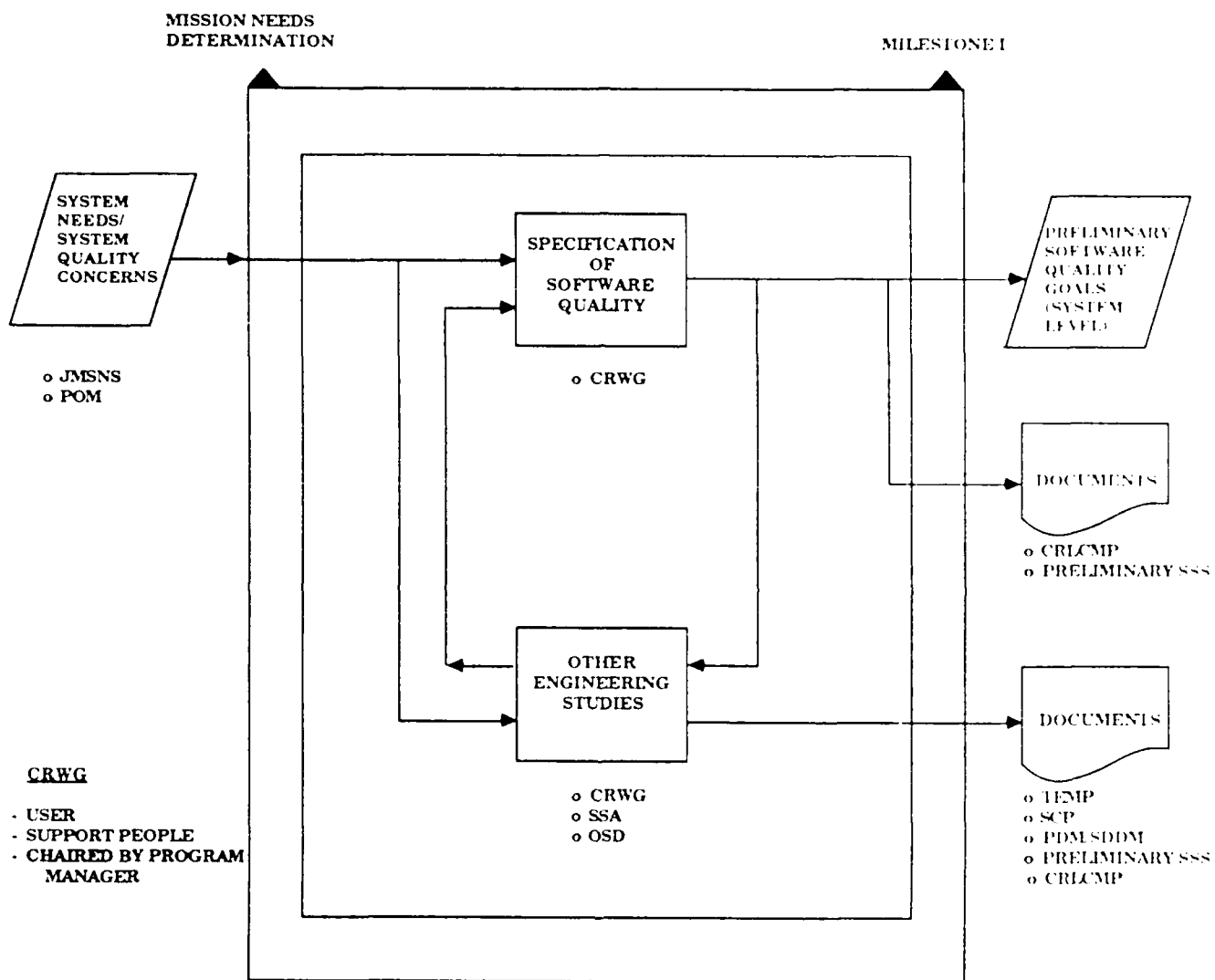


Figure 3.4-1. Specification of Software Quality During The Concept Exploration Phase.

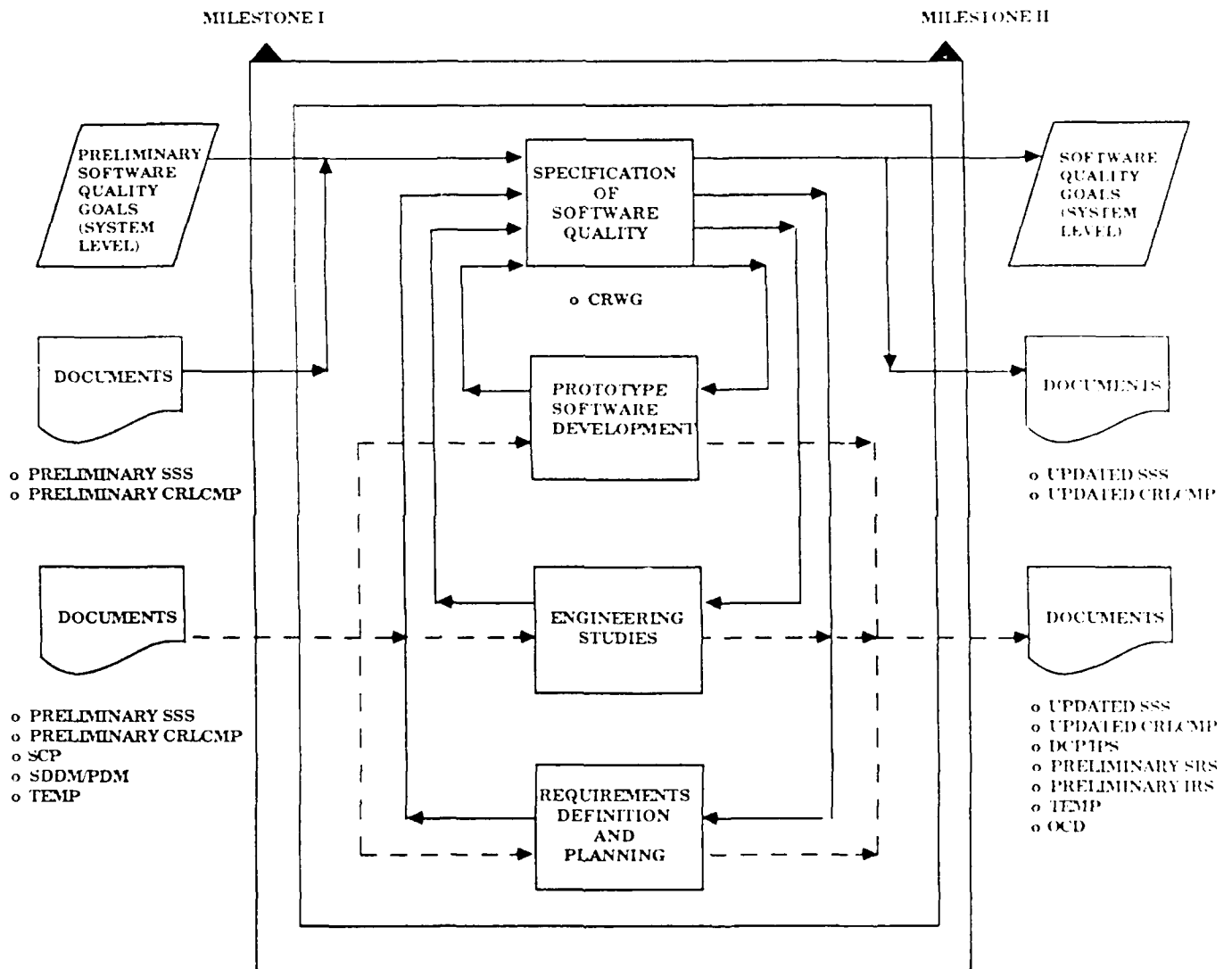


Figure 3.4-2. Specification of Software Quality During The Demonstration and Validation Phase.

Independent Validation and Verification (IV&V) personnel use the Assistant to obtain information about the specified factor goals and framework tailoring. IV&V personnel can also use the Assistant as an aid in assessing compliance. The reports automatically prepared by the assistant can then be forwarded to the acquisition manager along with other IV&V-type reports.

After a contract has been awarded, a development contractor can use the Assistant to assess and evaluate the factor goals. This provides the contractor with the capability of developing their own interpretation of the decisions/information leading to the goals. Differences in interpretation of the goals should be resolved in a series of meetings and recorded as history as soon as possible after contract award.

3.4.2 Relationship Between DOD-STD-2167A and the Assistant

When researching operational concepts, the relationship between the Assistant and DOD-STD-2167A was examined. DOD-STD-2167A presents one problem, since Paragraph 10.2.5.6 of the Software Requirements Specification (SRS) states that quality factor requirements should be specified for each Computer Software Configuration Item (CSCI). The Rome Air Development Center (RADC) Software Quality Specification methodology, on which the Assistant is based, states that quality factor requirements should be specified for each identified function.

In this context, the word 'function' may be misleading. The Assistant, and the RADC methodology do not prevent the user from identifying functions that directly correspond to CSCI's. This way the conflict can be resolved.

3.4.3 Relationship Between Proposed DOD-STD-2168 and the Assistant

Proposed DOD-STD-2168 mandates the formation of a quality evaluation plan. Details of this plan are documented in the Software Development Plan (SDP) or a separate Software Quality Program Plan (SQPP). Since the Assistant is a very useful tool for specification of software quality prior to Full Scale Development, the activities associated with proposed DOD-STD-2168 will normally take place after those of the Assistant. However, if both are performed concurrently, no conflicts exist. Use of the Assistant should be documented in the SQPP or SDP.

3.5 Support Environment

If multiple systems are deployed at various government or industry locations, then a combination of centralized and local support is envisioned. Local changes to the Assistant's knowledge base will be performed by a local system administrator. In order for the Assistant to gain knowledge and grow, information (in the form of rules, rationale, and facts), should be periodically extracted from all operational Assistant knowledge bases. Permission to extract this information should be established as part of the agreement to use the Assistant. This information should then be assimilated, filtered, and combined into one knowledge base. This process is optimally administered by a centralized support methodology, where a government agency, or government representative, such as the Data Analysis Center for Software (DACS/RADC/COED), collects and assimilates the information and sends updates to all active sites. Procedures and forms for extracting this information should be established by the administrative agency/representative. Ideally, this organization would also be responsible for maintenance of the Assistant.

3.5.1 Hardware Support

Since all hardware is commercial-off-the-shelf, support necessary for maintenance of system hardware components can be handled by the original vendor.

3.5.2 Software Support

For commercial-off-the-shelf system components, error fixes and updates will be handled by the original software vendor. Support for the developed software base (other than knowledge base enhancements) is to be determined.

3.5.3 Software Required

No additional software is necessary for the support and maintenance of the system.

3.5.4 Equipment Required

No additional equipment is necessary for the support and maintenance of the system.

3.5.5 Facilities Required

No customer facilities are necessary for the support and maintenance of the system.

3.5.6 System Administrator Requirements

In order for effective system support, a system administrator position should be established at each installation site. The system administrator is responsible for maintaining system security, by setting up accounts and passwords for the Assistant. The system administrator is also responsible for assuring that users cannot alter specification data without authorization. In addition, administration of both centralized and local changes to the knowledge base should be handled by the system administrator.

4. SYSTEM FUNCTIONS AND CHARACTERISTICS

The system functions provided by the Assistant to support the specification process are described in Section 4.1. Section 4.2 describes general functional capabilities that support a variety of the major system functions. User interaction is described in Section 4.3.

4.1 System Functions

The major system functions, shown in Figure 4.3-2, are described in Sections 4.1.1 (Rank Factors And Criteria), 4.1.2 (Quantify Factors), and 4.1.3 (Assess Compliance) respectively.

4.1.1 Rank Factors and Criteria

To rank factors and criteria the Assistant guides the user with the following subprocesses:

1. Identify application/functions
2. Assign factor and criteria rankings
3. Revise rankings based on interrelationships
4. Revise rankings based on costs.

These steps are illustrated in Figure 4.3-3. Step 1 is designed to help the user select an application area and decompose the system into a usable set of functions, for which quality requirements can be specified. Step 2 helps the user decide on the importance of the factors and criteria by taking into account quality considerations from a wide range of areas, such as: environment, application, and development characteristics. The final two steps are designed to help the user update the initial rankings, if necessary, based on considering the interrelationships between the factors (e.g., shared criteria), and the cost of achieving the specified factor rankings (Step 4). The revision of rankings based on costs includes an assessment of the cost of the rankings.

4.1.1.1 Identify Application/Functions

The purpose of this step is to identify each function which has separate quality requirements (See Section 4.1.1 of Volume II). When identifying the functions, certain guidelines regarding cost, level of required resolution, and logical system decomposition should be considered.

If too many functions are identified then the evaluation effort becomes excessively costly. Identification of too few functions may not provide the necessary resolution needed to isolate function-specific quality concerns. Most documentation for software systems is functionally oriented. To promote the most cost-effective evaluation, the functions should be identified following the existing system decomposition. If specification of quality starts very early in the life cycle, then goals are typically specified for the entire system since details regarding functional decomposition may not yet be known. Later, as more information about the system becomes known, goals can be allocated to the functions. The Assistant supports these issues by allowing the creation of new functions, and decomposition of existing functions, anytime during the system life cycle. These features are necessary to allow the user to address specific quality issues which arise during development/refinement.

All projects within a mission area and software type share similar function decompositions. This allows user to browse similar system specifications to assist in building new specifications. A generic system provides a baseline (model) from which to copy characteristics that are applicable to a new system. Changing functions is available for future use by browsing the system. In order to change the function decomposition structure the user can:

- o Change a function name
- o Combine two functions
- o Delete an existing function, or
- o Add a new function.

The Assistant supports a hierarchical function decomposition methodology. When a new function is added, it can either be added at the same level as an existing function, (it becomes a sibling), or added as a child (subfunction) of the existing function. To add a new subfunction the ADD TO CHILD operator is used. Figure 4.3-8 shows a context in which this operator can be used. If the user desired to add a new sibling to the functions shown (i.e., TOOLS through HARDWARE), the user would select ADD CHILD and indicate the appropriate parent in the hierarchy.

The results of changes to a function decomposition are available for future use. The user may select the system for browsing to review the updated function decomposition. The changes are not automatically incorporated into the mission area characteristics (i.e., the generic system), since this would require the Assistant to determine which changes are sufficiently general to apply to the mission area as a whole. Changes to the mission area characteristics based on specifications of individual systems can be made by the support agency (See Section 3.5).

4.1.1.2 Assign Quality Factors and Rankings

In this step, quality factors along with corresponding factor rankings are established for each function. In order to determine factor rankings (i.e., (E)xcellent, (G)ood, (A)verage) important information is assimilated from a wide range of areas, such as:

- o Application characteristics
- o Environment characteristics
- o Software development plan characteristics
- o System factor data
- o Quality survey data
- o Application specific quality concerns
- o Complementary factors.

Each of these areas provide different information indicating the importance of the factors. The Assistant shows the user only information that is relevant. For many of the areas, baseline data exists based on the mission area, and software type. Through a menu structure and the use of the mouse, users can select either YES or NO to the relevant questions. The system also supports the entry of a rationale with each response to facilitate explanations. In addition, the answer to any question can be changed at any time, and the Assistant will automatically adjust its conclusions. At any time, a user may achieve maximum flexibility by making selections from the complete list of questions through the use of the appropriate menu selection.

The Assistant will explicitly ask about the characteristics in an appropriate order. The questions asked will be relevant to the context determined by previous answers to the questions.

To assign the rankings to the factors, an algorithm is used combine the data from all of the areas. The basic philosophy behind this methodology is: the more areas that imply a factor/criteria, the more evidence that the factor/criteria is needed for the function. When data from multiple areas exist as a result of similar reason(s), the Assistant takes this into account and combines the data accordingly (See Section 4.1.1.2.8).

4.1.1.2.1 Application Characteristics

Application characteristics are system characteristics that are usually implied because of basic characteristics of the mission area and software type. (e.g., if the mission area is 'Command Control and Communications' then the existence of a communications network is implied [SOFTST]).

The Assistant displays only those application characteristics that might be applicable to the system. The user then has the opportunity to support or disagree with the characteristic by entering an answer of YES or NO. When an actual answer is provided by the user, the confidence in the inferred characteristic becomes 1.0 (highest), unless the user specifically enters a confidence other than 1.0. Inferred answers that the user has not reviewed generally have a lower confidence. The application characteristics can be reviewed in the form of a menu for revisions, changes and deletions of the responses.

The application characteristics are also used to infer other characteristics about the system, and to help automate worksheet tailoring during Quantify Factors (See Section 4.1.2). For instance, if the the system does not have any type of network, then metric worksheet elements regarding network protocols will not be applicable. The results of tailoring are documented in the tailored framework report (See Section 4.2.1.4).

4.1.1.2.2 Environment Characteristics

Environment characteristics of the system are usually implied because of interactions between the system and the environment in which it resides (e.g., if the system's deployment environment is on-board an aircraft, then Efficiency is important).

The Assistant displays these characteristics in the form of a menu. The data for this menu is an expansion of Table 4.1.2-2 in Volume II. By simply "clicking" the mouse, the user has the option of entering either a YES or NO answer. Rules relate environment characteristics with their implied quality concerns (factors). One use of this information is to help assign the factor and criteria rankings.

4.1.1.2.3 Development Characteristics

Development characteristics of the system are implied because of the development methods used. Some examples of development characteristics are:

- o Whether or not DOD-STD-2167A is used
- o Whether or not a high-order language is used
- o Whether or not a requirements language is used
- o What languag(es) is/are being used.

The information resulting from development characteristics will be used by the Assistant to infer information for a wide variety of areas. For example, if some documents or sections required by DOD-STD-2167A are tailored out, then the Assistant will automatically tailor out the appropriate worksheet questions. Likewise, if Ada is used then certain worksheet questions will have guaranteed values.

4.1.1.2.4 System Quality Factors

System quality factors are similar to software factors, but affect the system as a whole. Most system descriptions contain system-level requirements in terms of Availability, Reliability, Safety, etc. Since software is one component of an entire system, system-level requirements also affect software requirements. For instance, it is very hard to achieve a high system Reliability rate if the software Reliability rate is not high.

Volume II discusses translating system quality factors to software factors (See Table 4.1.2-3). The Assistant supports this type of relationship by automating Table 4.1.2-3. If a system description exists which contains system factors then these can be entered into the Assistant. The Assistant will then automatically translate these to the relevant software factors. This information is then used to help assign the initial factor rankings.

Since quality specification should be used even before a system description exists, the Assistant also supports the reverse of this relationship. Information about software quality rankings can be automatically translated into system factors. This information can help the user decide what the system factor goals should be from the perspective of software.

4.1.1.2.5 Survey Results

Factor surveys are another input to the factor ranking process. Surveys are one way that a wide range of people associated with the project can have an influence on the software requirements.

Data regarding survey results are entered into the Assistant. The Assistant then assimilates all the survey data and uses this information in the factor ranking process. People associated with different aspects of the project will have widely varying views on software quality. For instance, it can be expected that a test engineer is going to rank Verifiability very high but may not care a great deal about Portability. In order to achieve unbiased results, information regarding the job classification of survey recipients will be used by the Assistant in assimilating survey results. The Assistant will weigh factor rankings related to a survey recipient's area of specialization more highly than other factor rankings.

4.1.1.2.6 Specific Quality Concerns

Specific quality concerns are low-level details about the system that the user may know. In many instances the user may know about specific quality concerns and not about the factors. For example, the user may know that the system will have to support two different data base management systems, but not know that this implies that the criterion Application Independence and the factor Reusability are also applicable.

The Assistant will let the user choose specific quality concerns. The Assistant will use this information to contribute to inferences about the importance (rankings) of the factors and criteria, tailoring the framework, and quantifying the factor rankings (See Section 4.1.2).

4.1.1.2.7 Complementary Factors

As shown in Table 4.1.2-6 of Volume II, Reliability, Correctness, Maintainability, and Verifiability are all in a complementary relationship with other factors. Low quality rankings for these factors increase the difficulty of obtaining high rankings for the complementary factors. For example, a high Reliability score is difficult to achieve when Correctness and Verifiability are low.

When specifying factor rankings, the Assistant will automatically take complementary factors into account by adding rankings for additional factors when necessary, or re-adjusting certain factor rankings. At any time, the user can review all decisions made by

the Assistant regarding complementary factors, and change the results if desired. Changes are recorded by the Assistant.

4.1.1.2.8 Combining Selections to Determine Rankings

To assign the importance rankings to the factors and criteria the Assistant utilizes an algorithm to combine the data from all applicable areas. This algorithm is based on a heuristic combination of importance for the supporting reasons. Each reason implying the use of a factor or criteria is given an importance ranking.

The reason is based on the confidences in the premises that contributed to the reason and on the inherent confidence in the underlying rule. The resulting confidence in the reason is the product of the confidence in the least confident premise with the confidence in the rule. Reasons implying the use of a factor or criteria are combined by decreasing the distance to complete confidence (1.0) by an amount proportional to the confidence in the reason.

4.1.1.3 Consider Interrelationships

Factor interrelationships fall into three basic categories:

- o Shared criteria
- o Positive factor interrelationships
- o Negative factor interrelationships.

Interrelationships among the factors are explored to assess the technical feasibility of achieving the initial factor rankings. The assignment of different combinations of factors to a function can have either a beneficial or an adverse effect on the feasibility of achieving the desired quality levels. During this step, the Assistant weighs all three types of interrelationships and recommends either raising or lowering certain factor rankings.

The user can review the Assistant's recommendations for raising or lowering a factor ranking (See Figures 4.3-25 through 4.3-28). If the user disagrees with the underlying relationship (i.e., the communicativeness of usable software aids in verifying software) then they can indicate this by a negative response. For relationships in which potential counterexamples are known, they are incorporated into the knowledge base to help guide the user. Whenever a user disagrees with a relationship (either through a direct response or through a counterexample), the Assistant

automatically takes this into account and recalculates the recommended changes to the rankings.

The Assistant automatically updates the rankings based on positive relationships. Positive relationships provide an additional reason for needing a factor or criteria (i.e., to aid in achieving other factors or criteria). Rankings are not automatically updated based on negative relationships. Although negative relationships indicate a need to review the underlying reasons for needing contradictory factors more carefully, it is not the case that a negative relationship implicitly reduces the need for a factor.

4.1.1.3.1 Shared Criteria

A shared criterion is one that is an attribute of more than one quality factor (See Table 3.2-1 of Volume II). For factors that have many criteria in common, such as Maintainability and Verifiability, the difficulty of achieving high goals for both factors is reduced. If one evaluated factor score is high, then the scores for other factors which share criteria will also be high. Conversely, if an evaluated factor score is low, then the scores for other factors which share criteria will also be low. The Assistant takes shared criteria into account when considering costs (See Section 4.1.1.4).

4.1.1.3.2 Positive and Negative Factor Interrelationships

Positive factor interrelationships occur when cooperative or beneficial relationship(s) exist between factors. For example, Usability is in a cooperative relationship with both Maintainability and Reliability. The underlying rationale is that operability of usable software aids in software verification and maintenance (See Table 4.1.3-2 of Volume II).

Negative factor interrelationships occur when an adverse relationship exists between factors. For example, the need for high Reliability adversely affects Efficiency. The underlying rationale is that the additional code required to guarantee high Reliability increases run time and storage requirements (See Table 4.1.3-3 of Volume II).

The Assistant assigns a "degree of effect" rating for each positive or negative factor interrelationship. This rating is used later when the Assistant combines all types of relationships and recommends increasing or decreasing certain factor rankings.

4.1.1.4 Consider Costs

Relative costs associated with using software quality technology are explored to determine the feasibility of achieving the initial factor rankings. Costs associated with the following areas are explored:

1. Specifying quality requirements
2. Allocating the requirements to the design
3. Designing and building quality into the product, and
4. Evaluating the achieved quality levels.

The use of the quality methodology results in an increased awareness of quality throughout the life cycle. In addition, costly problems occurring late in the system life cycle are avoided by using the methodology to build higher quality levels into the system during requirements, design and development.

The assignment of different combinations of factors to a function can have either a beneficial or an adverse effect on the relative costs during different life cycle phases. Based on information provided in Tables 4.1.4-1, 4.1.4-2 and 4.1.4-3 of Volume II, the Assistant assigns a cost estimate to each factor, and updates these cost estimates by taking factor interrelationships into account. The Assistant then considers costs and identifies factors that are high cost drivers and recommends imposing cost limits for certain factors.

4.1.2 Quantify Factors

This section describes the process of tailoring the quality metrics framework to a selected system/function and determining numerical values for the factor goals based on that tailored framework.

The metrics framework for the Assistant is the RADC metrics framework. Based on the answers to acquisition questions, user-specified and inferred, the Assistant is able to tailor the framework to be specific to a selected system/function.

As an aid to describing the relationship between the acquisition questions and concerns and the metrics of the quality framework, the metric-elements of the framework are categorized as follows:

- o Application metric-elements -- The value of the metric is inherent to the problem. (e.g., Does the time to perform successful synchronization impose constraints upon system computation or response time ?)
- o Approach metric-elements -- The value of the metric is inherent to the development process. (e.g., Is a HOL used ?)
- o Execution metric-elements -- The value of the metric depends on the properties of the actual mapping of the problem to the machine. (e.g., How many branches ?)

4.1.2.1 Acquisition Questions

The answers to acquisition questions concerning application, development, and environment characteristics are used to determine the relevance of lower-level acquisition questions.

The answers to lower-level questions are used as a basis for the following inferences:

- o To infer the answers to other lower-level questions. (e.g., If the application is a Flight Control System, then with a high confidence the system is mission critical and on-board.)
- o To infer the relevance of other lower-level questions. (e.g., If DOD-STD-2167A is used, then issues relating to tailoring DOD-STD-2167A become important.)
- o To infer the use of a more specific set of baseline values for the metric-elements. (e.g., If DOD-STD-2167A and Ada are used in the development of the system/function, then the set of similar systems/functions is restricted to those that were developed using a similar approach.)
- o To infer counts for related data items. (e.g., If Ada is used, then the control variable for a FOR LOOP cannot be used outside the scope of that LOOP.)
- o To infer tailoring of the metrics framework. [e.g., If the system is mission critical, system is on-board, and human is on-board, then the system must continue to operate in real-time fashion to a combat situation where physical damage to the system can occur and human life can be lost. This results in Survivability depending more on Reconfigurability than Anomaly Management (physical damage more likely than logical problem), more on Anomaly Management than Autonomy (need to recovery from faults more likely than need to provide alternative functions or interfaces), and more on Autonomy

that Modularity (need to use interface involving human override more likely that the need to fix or replace segments of the software].

An example of relationships between asking questions and inferring results is shown in figure 4.1-1. The double bar arrow indicates an inference and a single bar arrow indicates initiation of a question based on a fact in the data base. In this example, several inferences are made based on knowing that the application is a flight control system (i.e., the system is mission critical, 2167A and Ada are used, etc.). Once it is known that 2167A is used, then questions about how 2167A is tailored are initiated. Although this example is an oversimplification (i.e., many of the inferences shown are made with a confidence factor less than 1.0), it illustrates how the use of an inference mechanism controls the asking of relevant questions.

The user of the Assistant is able to review and change the user-specified and inferred answers to questions. Of course, if the user changes an inferred answer, then the new answer is considered user-specified.

The user of the Assistant is able to review and change the user-specified and inferred tailoring of the metrics framework. Of course, if the user changes an aspect of the inferred tailoring, then that aspect of the tailoring is considered user-specified.

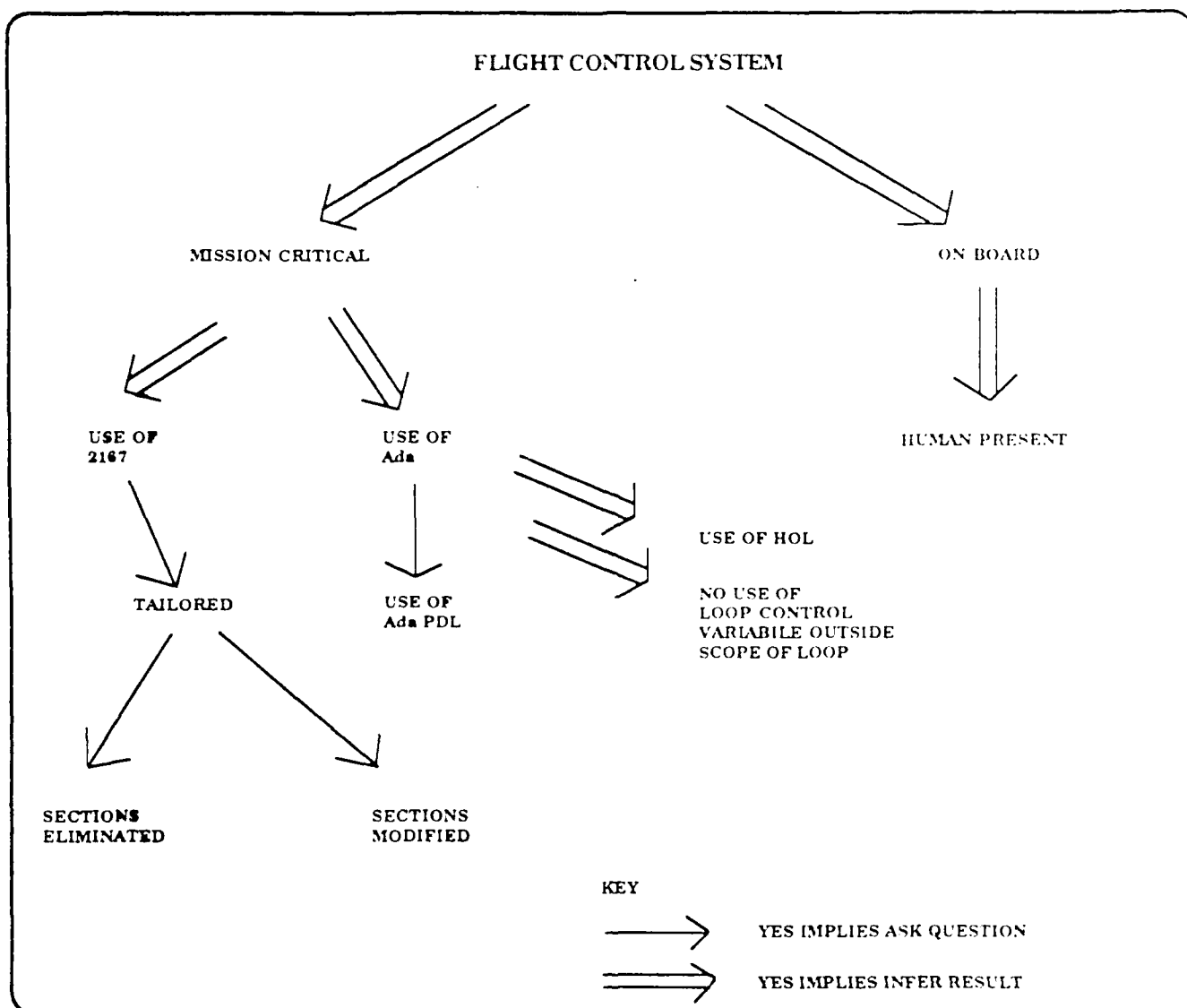


Figure 4.1-1. Inferences Based On Answers To Questions.

4.1.2.2 Baseline Values

Baseline values for the metric-elements are established as follows:

- o For application metric-elements, application specific baseline values are determined based on the typical attributes of that application. (e.g., If the system is a Flight Control System, then with a high confidence the system is a real-time application.)
- o For approach metric-elements, approach-specific baseline values are determined based on the typical attributes of particular aspects of that approach. (e.g., If the system is being developed using a RATIONAL Computer, then with a high confidence the system is being developed using an HOL, namely Ada.)
- o For execution metric-elements, baseline values are determined based on data derived from actual coded units (i.e., from previous applications of the quality technology). When possible, the values are specific to the particular application and approach.

In order to compute expected scores during Quantify Metrics (Figure 4.3-37), baseline values for metric-elements consist of the following components:

- o An upper bound for the majority of projects.
- o A lower bound for the majority of projects.

where each of these values is synthesized by quality researchers from historical data for software with attributes similar to those of the chosen system/function. From a set of baseline values, an anticipated value range is calculated for each metric of the framework.

The user of the Assistant is able to specify baseline values for any subset of the metric-elements of the framework. A baseline value for a metric-element consists of a single component that indicates an anticipated lower or upper bound for that element. Baseline values are used for establishing expected ranges for metric, criterion, and factor scores.

The Assistant is able to infer baseline values based on answers to acquisition questions. (e.g., If 2167A is used, then baseline values might be inferred based on the coding standards associated with 2167A.)

From a set of baseline values, an anticipated lower bound value is able to be calculated for each metric of the framework. For example, the lower bound could be calculated as a minimum of the set of baseline values.

4.1.2.3 Data Item Counts

The user of the Assistant is able to specify actual counts for any subset of the data items of the metrics framework.

The Assistant is able to infer counts for data items based on answers to acquisition questions. (e.g., If Ada is used, then the control variable for a FOR LOOP cannot be used outside the scope of that LOOP.)

A data item count consists of a single component indicating the number of occurrences of that item. From a set of data item counts, an expected value is able to be calculated for each metric of the framework.

4.1.2.4 Calculations

Quantified factor goals for a system/function are calculated using the metrics framework after tailoring for that system/function.

Quantifying factor goals for a system/function involves using a combination of the following kinds of numeric information:

- o User-specified counts for the data items
- o Inferred counts for the data items
- o User-specified baseline values for the metric-elements
- o Inferred baseline values for the metric-elements
- o Baseline values for the metric-elements based on data items for the system/function
- o Baseline values for the metric-elements for similar systems/functions
- o Baseline values for the metric-elements for the specified mission area
- o Baseline values for the metric-elements for any general application

where the precedence for using a particular kind of numeric information is indicated by its position in this list. Being at the top indicates highest precedence and being at the bottom indicates lowest precedence.

4.1.3 Assess Compliance

The assessment of the compliance step uses the evaluation results of Volume III (Software Quality Evaluation Report) as input and produces a report assessing the degree of compliance with specified quality factor goals. The compliance step is performed by the individual(s) responsible for the initial goal specification. It is performed long after the first specification process, during each software development phase.

The Software Quality Evaluation Report includes the results of data collection and reduction, analyses of variations from factor goals, analyses of criterion, metric, and metric-element scores which detract from the achieved scores. This evaluation is usually performed by individual(s) other than those responsible for the quality specification. Many of the steps in this process are supported by the QQuality Evaluation System.

The Assistant supports the assessment of compliance step as performed by the acquisition manager. The acquisition manager uses the Assistant and the evaluation results for determining:

- o 1) Whether or not the quality factor goals (requirements) were unrealistic (reviewing the goals and rationale, what-if analysis) and
- o 2) Whether or not the contractor has neglected to meet quality requirements (comparisons to expected scores and thresholds for individual criterion, metrics and metric-elements)
- o 3) Whether or not results were biased by the scoring method used (what-if analysis can be used to determine the effect of tailoring on scoring)
- o 4) Quality trends as compared to the specified goals.

Since the development or IV&V contractor usually know specific details about the project, any recommendations presented in the evaluation report should be carefully considered during this step. For example, the rigid enforcement of excessively high factor goals, may cause project schedule and budget slips.

Although both Volume II and III describe assessment of compliance steps, the emphasis is different. Volume III emphasizes the evaluation results and their relationship to the specified goals. Volume II emphasizes determining an appropriate acquisition response to scoring variations. Therefore, the Assistant requires the capability to review the evaluation results in the context of the quality goals and rationale. Notice that each of the four points above requires detailed information from the specification process. For example, the user may need to redo some of the specification steps in response to evaluation results (that is, in order to determine whether to change goals, or to directly change goals). This is the principal distinction between the assessment of compliance steps supported by the Assistant versus QUES.

To perform the above activities the Assistant automates the following features:

- o Input of the QUES revised project structure.
- o Input of the actual evaluation results
- o An assessment of compliance graphic report.

4.1.4 System Administration Functions

Certain types of information within the Assistant's knowledge-base are vital to effective operation. For example, changes to the baseline metric element values, system specific information, and supporting tables can have drastic global effects on the Assistant's operation. In order to maintain system security and integrity, a system administrator is responsible for:

- o Providing user access to mission areas and projects
- o Adding and removing users to/from the system
- o Entering observed quality factor rates (see Table 3.1-2. Quality Factor Ratings of the Guidebook), and software quality management indicator information [SOFTIND].
- o Modifying rules, baseline data and other supporting information.

The system administrator functions can be entered from the top level roadmap by a user with a privileged account and password. Under the system administration menu, the only operations that an unprivileged user can perform are 1) Changing his/her own password, and 2) Changing the access privileges to projects for which he/she is the owner.

4.2 Computer System Functions

This section describes general functional capabilities that support a variety of the major system functions presented in Section 4.1.

4.2.1 Reports

This section describes the reporting capabilities supported by the Assistant.

4.2.1.1 Snapshot Report

The Assistant is able to produce anytime during a session a hard-copy report of the information shown on the immediate screen. This allows the user to produce hard-copy documentation based on the information displayed by the Assistant.

4.2.1.2 Goals and Rationale Report

The user is able to review the factor/criteria goals and the associated rationale for any system/function.

The Assistant is able to produce a hard-copy report of the factor/criteria goals and the associated rationale for any system/function in a form suitable for inclusion in a Software Requirements Specification.

4.2.1.3 Questions and Rationale Report

The user is able to review the answers to the relevant questions for any factor of a system/function. The user is able to ask for a rationale to an answer, add to the rationale, or change the answer.

The Assistant is able to produce a hard-copy report of all relevant questions for any factor of a system/function where

- o if the answer is user-specified, then reported are the question, the answer, and an indicator of whether the answer was supplied directly by the user or indirectly through copy of user-specified answer from another system/function.

- o if the answer is inferred by the Assistant, then reported are the question, the answer, and the rationale for answer, and the rationale for the relevance of the question.
- o if the question is unanswered, then reported are the question and the rationale for the relevance of the question.

4.2.1.4 Tailored Framework Report

The user is able to review the metrics framework for any factor of a system/function. The formula for calculating a quality metric is shown in terms of a weighted average of the related metrics in the next level down of the hierarchical framework. The user may change the weighted averages for any of the metrics reviewed.

The Assistant is able to produce a hard-copy report of the tailored metrics framework and rationale for any factor of a system/function.

4.2.1.5 Change Report

The user is able to review the answers, user-specified or inferred, that change during the current session for any factor of a system/function.

The user is able to review changes in factor goals for a system/function that occurred during the current session.

The Assistant is able to produce a hard-copy report of all the answers, user-specified or inferred, for any factor of a system/function during the current session.

4.2.1.6 Assessment of Compliance Report

In order to document the results of the assessment of compliance step, the Assistant produces a detailed report which documents the following information:

- o The calculated scores for each entity (CSCI, etc.) and function for each applicable development phase
- o Where the specified goals were different from the achieved
- o The achieved scores for framework elements that fall below the lower bounds of the specified goals will appear as RED on the assessor graph.

- o The achieved scores for framework elements that fall below the midpoint of the lower and upper bounds of the specified goals will appear as YELLOW on the assessor graph.
- o The achieved scores for framework elements that are higher than the midpoint of the lower and upper bounds of the specified goals will appear as GREEN on the assessor graph.
- o A summarization of any of these results by application, approach or execution (see Section 4.1.2).

To support maximum flexibility the level of detail of the assessment report will be user-defineable. By supporting this feature the Assistant is able to report the results of analysis down to the factor, criteria, or metric-element level. This added flexibility allows the Assistant to produce reports for all levels of management.

4.2.1.7 Percent Unknown Report

In order to allow a user to see how complete the specification for a particular function is, the Assistant provides a pie chart showing the following information:

- o The percentage of answered questions which have an answer of UNKNOWN, and
- o The percentage of answered questions which have an answer other than UNKNOWN.

This pie chart will be shown separately for each function of the system. It allows the user to determine the completeness of the specification for any function. Because the system allows results (goals/rankings/tailoring, etc.) based on a specification containing Unknown answers, this report is necessary to allow the user to determine how much information will have to be added to the specification in the future to improve completeness.

4.2.2 Reasoning

The Assistant performs reasoning (i.e., makes inferences) to build additional information from the facts about the system and system needs provided by the user. The reasoning is essential to the Assistant's ability to "fill in" information and avoid asking detailed questions that would be distracting to the user, could not be answered by the user, or would collectively require too much time to answer.

Inferences made by the assistant are assigned a confidence factor which indicates the confidence that the conclusion is correct given a set of premises, each with their own associated confidence values. Reasoning with confidence factors has been used successfully in a number of expert systems. The Assistant allows users to review all conclusions made and affect the results if desired.

The Assistant uses both forward-chaining reasoning (i.e., working from the data towards the goal) and backward-chaining reasoning (i.e., working from the goal towards the data). Forward-chaining is used in Rank Factors And Criteria (Section 4.1.1) to derive system and software characteristics and needs from the information provided by the user. For example, the existence of a communications network and a network protocol might be inferred from the application. These characteristics might in turn infer other characteristics or needs. The user will be given the opportunity to review these inferred characteristics. When the user reviews a characteristic, or selects or deselects a characteristic, its confidence factor is increased or decreased.

Backward-chaining is used in Quantify Factors to tailor the framework and determine expected scores based on the user's input. In order to tailor the framework, the inference engine tries to prove the hypothesis that the selected factor is applicable. In doing so, it resolves the applicability of each of the criteria, metrics and metric-elements comprising that factor. An example of some of the rules that would be used to tailor the factor Portability is given in Appendix A. In order to determine expected scores, the inference engine works backwards from the factor goal. This involves resolving scores for criteria, metrics, and possibly, metric-elements and data items.

4.2.3 Explanations

The Assistant allows the user to obtain explanations of the reasoning performed. These explanations contribute to the rationale for rankings, numerical goals, and tailoring as well as provide the user the opportunity to review and modify the conclusions made during reasoning. The explanations follow the approach used in TEIRESIAS [TEIRESIAS]. Examples of explanations are provided in Figures 4.3-32, 4.3-33, and Figures 4.3-26 through 4.3-28. In response to WHY, the Assistant backs up the goal tree and presents the rule used to derive the sub-node from its ancestor. In this way, it provides an explanation of why the Assistant wants to know that information (presented in the form: If A were concluded then B could be concluded). In response to HOW, the Assistant explains the nodes lower down in the goal tree that have been or will be expanded. In this way, it provides an explanation of how a fact would or could be concluded.

4.2.4 Knowledge Acquisition

The Assistant can help software quality experts formulate new rules to be added to the Assistant's knowledge base. This allows the expertise of the Assistant to grow with software quality expertise. Also, individual user sites can tailor the rules to their application and environment. The new or modified rules could later be incorporated into the baseline version of the Assistant if appropriate.

The knowledge acquisition features of the Assistant follow the approach used in TEIRESIAS [TEIRESIAS]. The Assistant's rules can be reviewed and modified as they are applied. The rules are presented in a stylized English form.

Modification of rules is limited to a user with system administration privileges. The modified rules form a new version of the Assistant. The Assistant supports multiple versions.

4.2.5 What-If-Analysis

Users are provided the capability to modify assumptions and conclusions, and view the impact of those changes on the conclusions of the various procedures and the final quality goals. For example, the user can modify the DOD-STD-2167A tailoring (e.g., add or delete a Data Item Description (DID) or DID paragraph) to determine the impact on the quality goals.

Although this kind of what-if analysis is an important feature of the Assistant, the functionality required to support it is minimal. What-if analysis is supported by allowing the user to freeze the state of the knowledge base and save it, modify an input to see the effects of the change, and restore the knowledge base to its frozen state.

4.3 User Interaction

The user interaction emphasizes minimizing the amount of "computer-ese" required on the part of users. Users of the Assistant are expected to have a wide range of familiarity with computers, including some with no computer experience. The intent is to provide user interaction that does not require memorizing special key sequences or command sequences. The user interaction is intended to be usable by first time users with only a very limited introduction.

Although it supports inexperienced users, the Assistant should not discourage experienced users as a result of clumsiness.

The essential elements of the operational modes are summarized below.

- o Roadmaps are provided to show the three high-level steps in the methodology, their purpose, and their relationships. A second level of roadmap shows steps within each of the three major steps.
- o The user can move to any step on the roadmaps by selecting that step. If previous steps have not been completed the Assistant fills in the information as much as possible based on inferences.
- o Within a step on a roadmap the user is offered a sequence of menus to select a path of operations to perform the step.
- o The user can return to the roadmap at any time. The roadmap shows the current step in the process.
- o Help can be selected to obtain a description of each selection on a menu.
- o Previous step can be selected to move to the previous sub-step within a given step on the roadmap.
- o Windows display information and can be scrolled to view more information than fits in the window.

The operational modes of the Assistant are illustrated in a sequence of figures which comprise a scenario representing a hypothetical session. The scenario is intended to illustrate the operational concepts of the tool to demonstrate what the Assistant can accomplish. The figures in this scenario are not intended to specify user interface details such as the placement of menus and windows, or the complete list of menu selections. These details are deferred to the Software Product Specification.

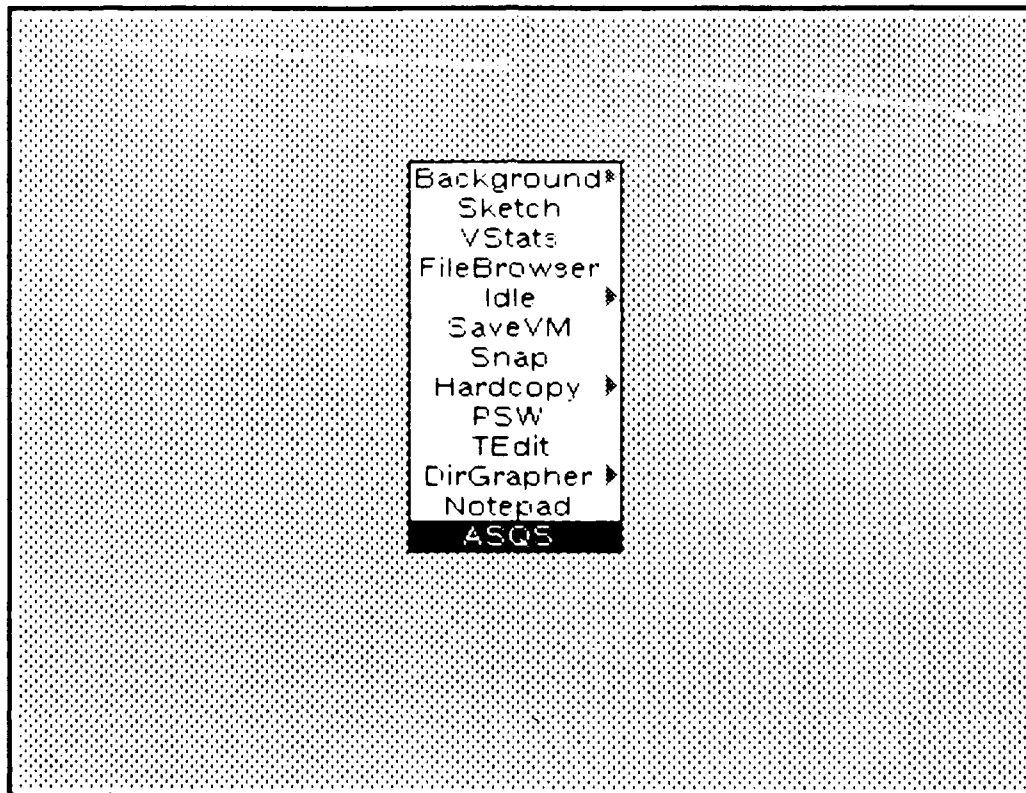


Figure 4.3-1. The Assistant Executes in a System Software Environment.

The Assistant executes in an off-the-shelf environment comprised of the computer system and associated operating system. Rudimentary functions such as file management are provided by this environment. For example, management of the software quality specification files is provided by the FileBrowser.

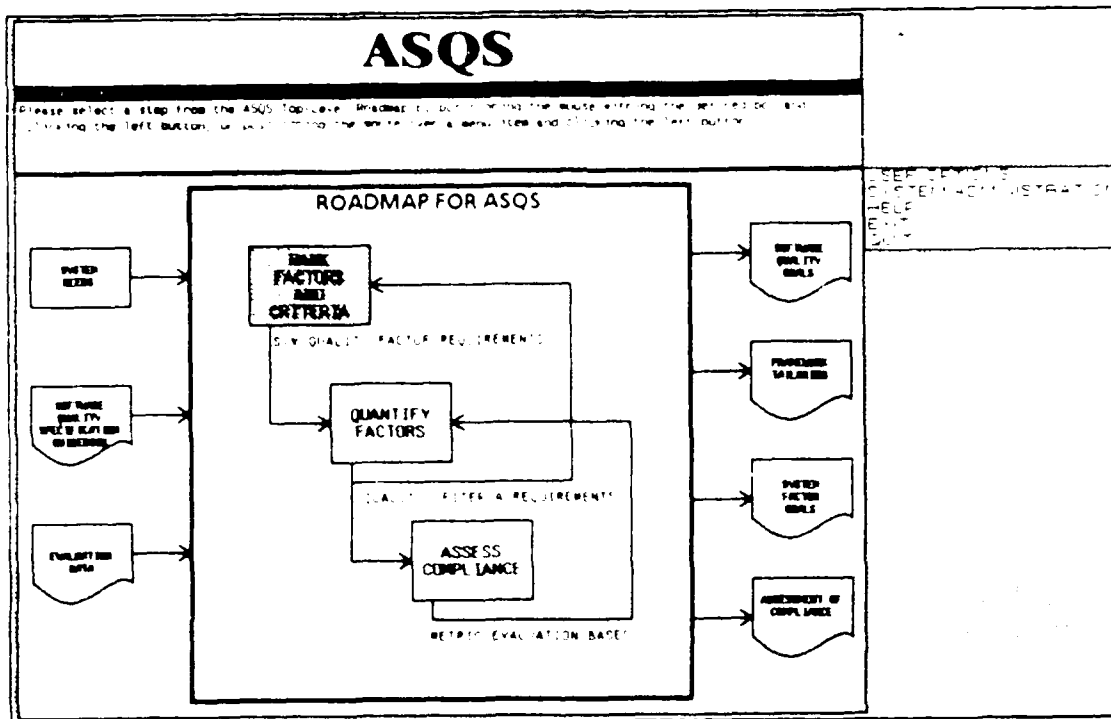


Figure 4.3-2. A Top-Level Roadmap Illustrates the Specification Steps.

The first screen upon entering the Assistant is a top-level roadmap. The roadmap shows the highest level of steps in the process of specifying quality. (An overview of the steps is provided in Sections 4.1.1, 4.1.2, and 4.1.3.) The user can move to any of the steps to perform the desired activity. If previous steps are skipped the Assistant extrapolates based on the available information and inferences. The roadmaps can be displayed at any time to show the current step or move to a different step.

The most recently entered step is shown by the shaded menu selection (Rank Factors and Criteria). Holding down the mouse on a given item in the roadmap allows users to obtain a description of the indicated item. Throughout this scenario we will assume that the shaded selection was selected by the user.

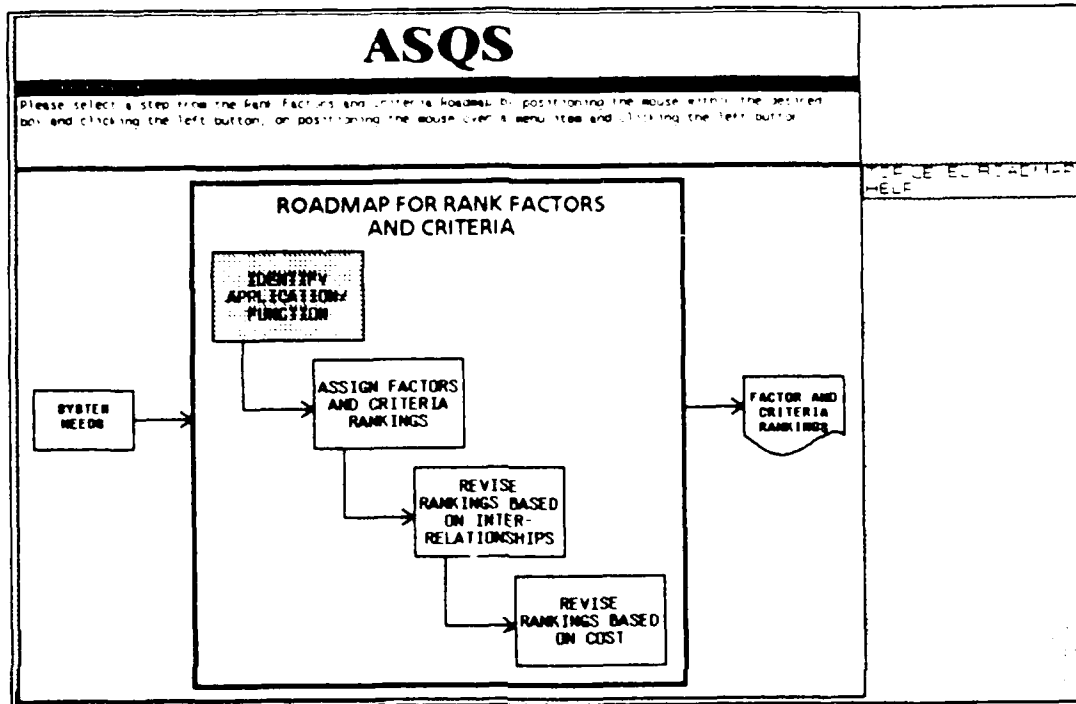


Figure 4.3-3. A Second-Level Roadmap Illustrates the Steps of Rank Factors and Criteria.

An overview of the steps of Rank Factors and Criteria is provided in Section 4.1.1.

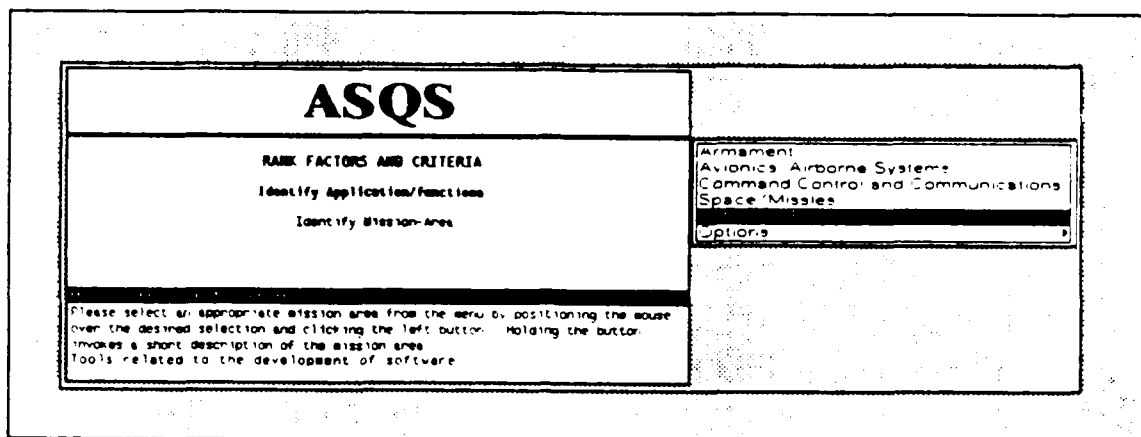


Figure 4.3-4. Menu Guide Users through the Sub-Steps within a Step on the Roadmap.

A menu-driven user interface provides flexibility for users to tailor use of the Guidebook to their needs without prior knowledge of the Guidebook procedures or information. This figure shows the step of selecting a mission area. Holding the cursor on a menu item provides a description of the indicated mission area. The screen background will change to reflect the newly selected mission area.

The mission areas are derived from the Software Test Handbook [SOFTST]. Software Development Tools is added to illustrate the scenario, which is for a Software Development Environment.

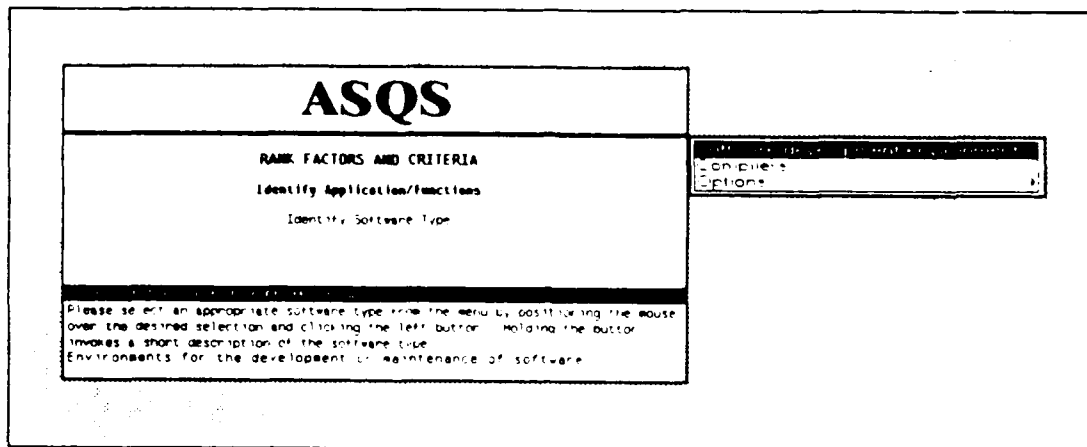


Figure 4.3-5. Selection of Mission Area and Software Type Allows Tailoring to User needs.

A knowledge base about the Mission Area and Software Type comes into play when the user selects them. For example, system characteristics and needs are automatically assumed. They can be reviewed and changed by the user.

The selection PREVIOUS STEP takes the user back to the previous sub-step within a step on the roadmap (e.g., here it returns to selecting the mission area).

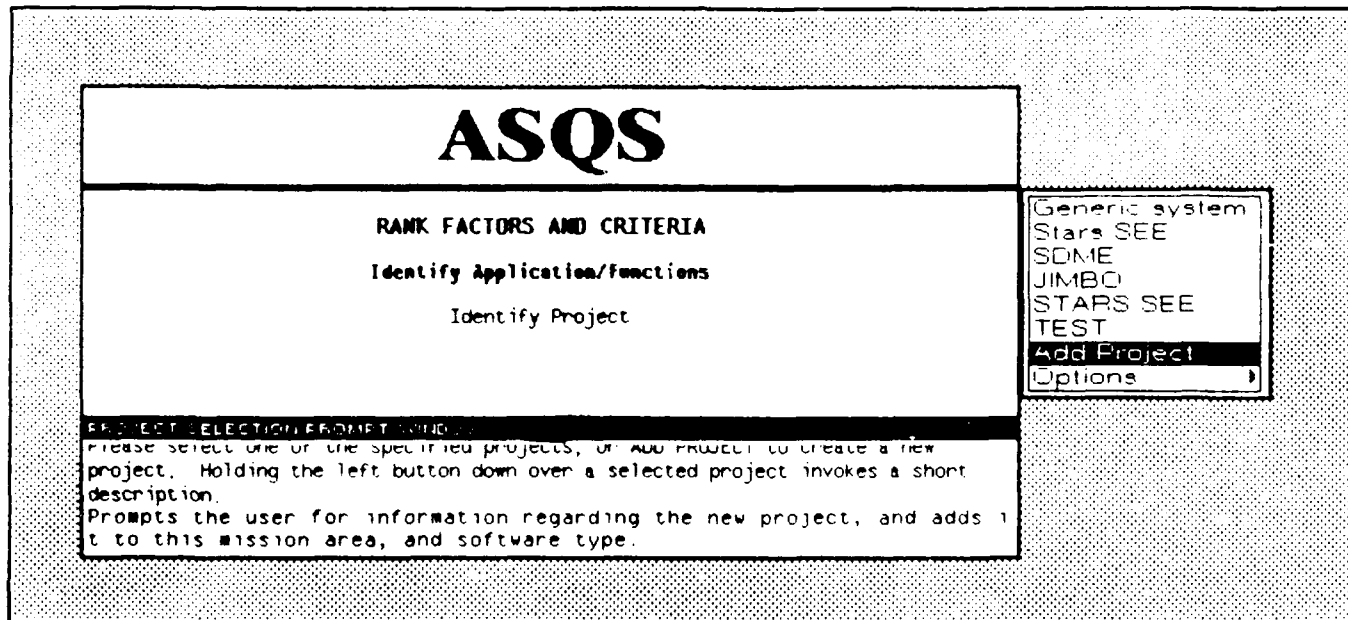


Figure 4.3-6. Examples Provide for Decision-Making Based on Experience from Other Programs.

In this scenario we illustrate the specification of quality for the Software Technology for Adaptable Reliable Systems (STARS) Software Engineering Environment (SEE) [SEEOCD]. The figure shows that the experience of specification from other software engineering environments (e.g., SDME) can be examined and utilized. Also, a generic system is available for review (the project GENERIC SYSTEM). This provides a specification for a software engineering environment in general. It can serve as a source of information (by copying characteristics) for the current project.

Add Project can be selected to add a new project as shown. In this scenario however, we will specify quality for the already existing STARS SEE project by selecting STARS SEE.

ASQS	
SYSTEM DECOMPOSITION FOR STARS SEE The Software Technology for Adaptable Reliable Systems (STARS) Software Engineering Environment (SEE)	VIEW DESCRIPTION CHANGE NAME CHANGE DESCRIPTION CHANGE WEIGHT DELETE DECOMPOSE ADD CHILD COMBINE COPY CHARACTERISTICS READ NOTE EXPLAIN VERSION MANAGEMENT OPTIONS CONTINUE
STARS SEE	

Figure 4.3-7. Specifying a New System Based on Past Experience.

By copying characteristics from another function (within the same project, or from another project or mission area) the user can build a specification with a previously developed specification as the starting point. This is appropriate when the new specification is for a similar system, an upgraded system, or an evolutionary system development. In this case, characteristics are copied from the GENERIC SYSTEM, which is appropriate when first beginning a new specification.

Once the characteristics are copied, changes to the other system specification do not affect the new system specification.

This same capability can be used to copy the specification for an individual system function specification from another system specification. A menu path (not shown) is provided to copy functions from other applications.

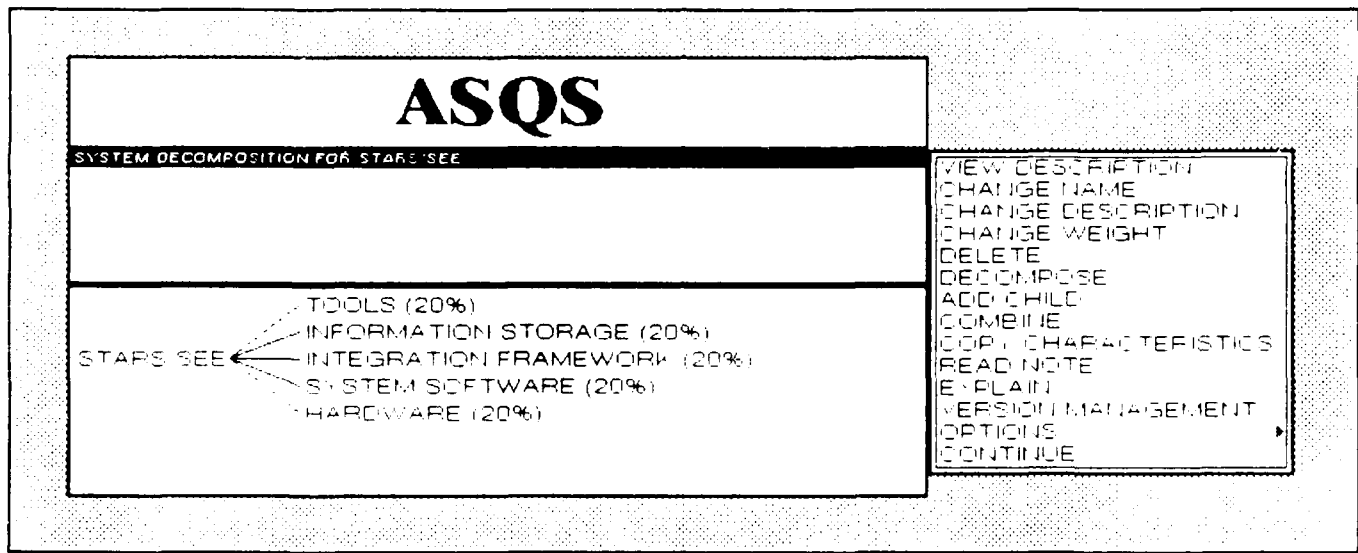


Figure 4.3-8. A User Can Move Through the Hierarchy of System and Software Functions.

Once the user has selected a system, the top-level function becomes System (i.e., overall system). By selecting MOVE TO SONS the user can move to a lower-level in the function hierarchy to allow specification of quality for the functions in the system. The function hierarchy allows users to specify quality to the level of detail desired.

The initial list of functions is derived automatically based on the application and other system specifications which the user may have incorporated.

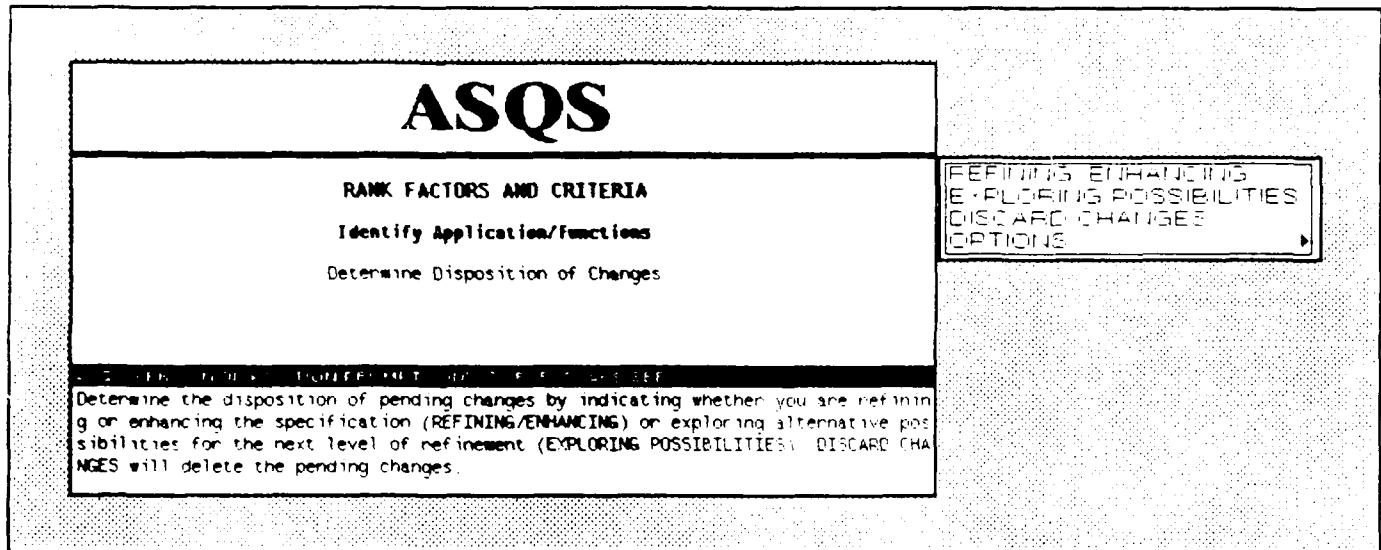


Figure 4.3-10. History Provides the Basis for Informed Decision-Making in the Future.

The Assistant records decisions made by the user, and provides explanations based on those decisions and (when applicable) based on its own reasoning. The explanations are crucial for making modifications when requirements or other circumstances change. The explanations also provide continuity across personnel changes. Establishing an easily accessible history of the quality specification provides the opportunity to learn from experience.

This slide shows the menu to determine the disposition of changes. REFINING/ENHANCING adds a new leaf to the version tree and sets the current version to the new leaf. EXPLORING/POSSIBILITIES adds a new leaf, but leaves the current version at the parent so that other possibilities can be explored.

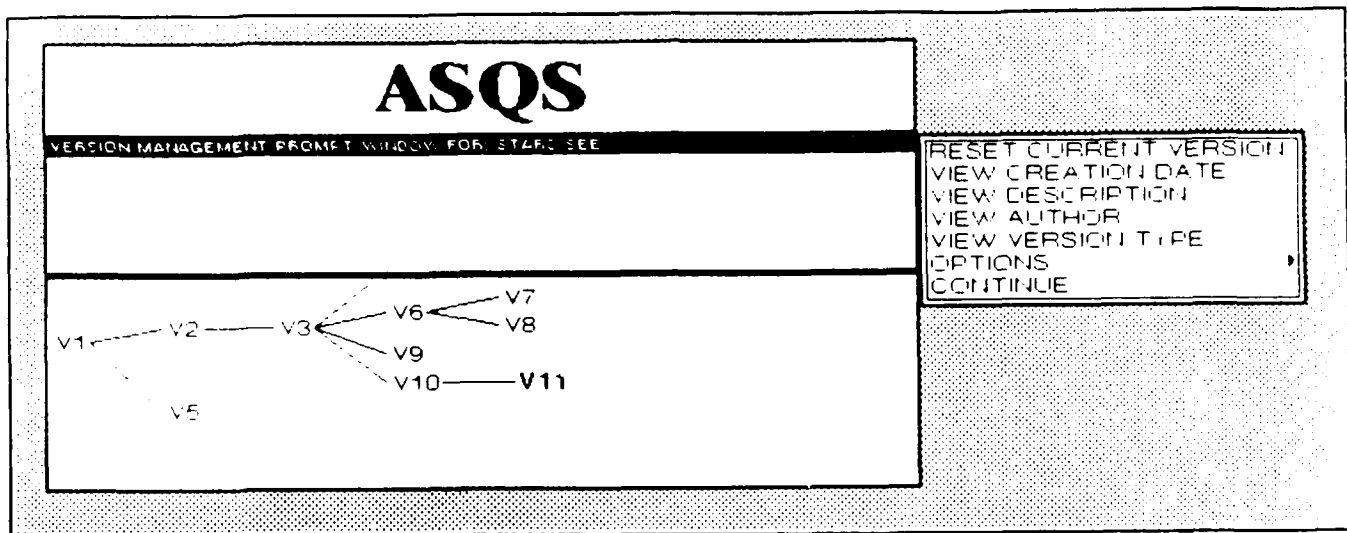


Figure 4.3-11. The Version Tree Provides Access to History.

Each version of the software quality specification is preserved in the version tree. The current version is marked in bold (V11). Version 11 represents a refinement to version 10. Versions 6, 9, and 10 represent alternative possibilities that were explored by the specifier. Another alternative derived from version 3 is outside of the window. This version can be viewed by selecting version 3 to center it in the window.

Information about each version can be reviewed by selecting the VIEW menu items. RESET CURRENT VERSION can be used to set the current version to other than the current one. CONTINUE returns the user to the roadmap.

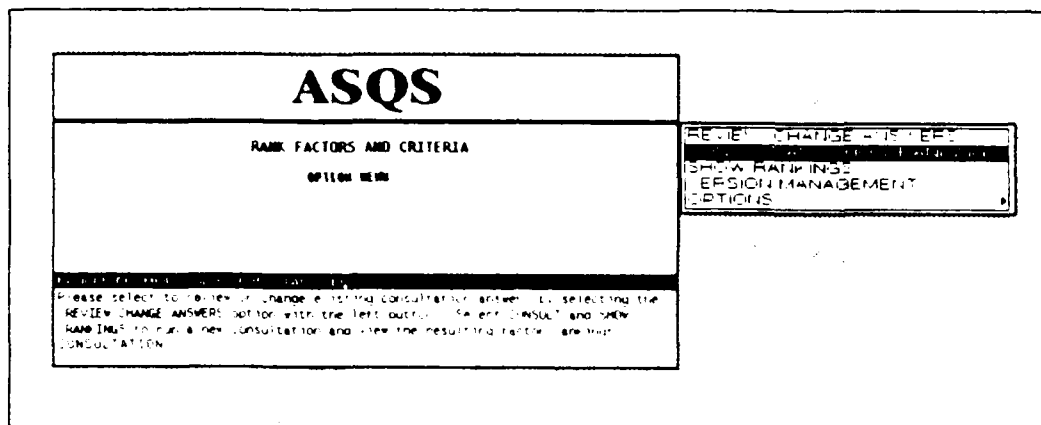


Figure 4.3-13. Optional Paths Allow Users to Specify as They Choose.

At this point the user can either work forward to completed rankings by browsing various quality concerns to select the ones that apply to the system (REVIEW/CHANGE ANSWERS), or work backwards by reviewing the current specification as it stands and isolating aspects and making changes (CONSULT AND SHOW RANKINGS). At any time, the user can return to this menu and choose to review the overall rankings reflecting the current state of the specification (SHOW RANKINGS). Ordinarily a user will use a combination of selecting and reviewing to build the specification.

Here the user chooses to review the current specification first since it is derived from the specification for a generic system.

<h1>ASQS</h1>														
Factor Rankings for STARS SEE														
<h2>RANK FACTORS AND CRITERIA</h2> <p>Assign Factor and Criteria Rankings</p> <p>SOFTWARE QUALITY FACTOR IDENTIFICATION FORM - INITIAL GOALS</p>														
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="transform: rotate(-45deg); white-space: nowrap;">SYSTEM OR SOFTWARE UNIQUE FUNCTION</div> <div>SOFTWARE QUALITY FACTOR</div> </div>	PERFORMANCE					DESIGN			ADAPTATION					
	EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	USABILITY	PORTABILITY	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	INTEROPERABILITY	PROTECTABILITY	RECOVERABILITY	
	STARS SEE	L		M		M		L	M	M	L	L		
	FUNCTION CAPAB	L		M		M		L	M	M	L	L		
	INFORMATION STORAGE	L	M	H	L	M			M	M	L	M	L	
	INTEGRATION FRAMEWORK	L		M	L	M		L	M	M	L	M	L	
	SYSTEM SOFTWARE	L		H	L	L	H		M	M		M	L	

How
Continue

Figure 4.3-14. The Specification is Reviewed for Tailoring to the STARS SEE.

The selection CONSULT AND SHOW RANKINGS results in a consultation dialogue between the user and the Assistant. Any questions which the user wants to defer can be answered UNKNOWN. The Assistant extrapolates relative factor rankings from the known information. "H" represents High, "M" is medium, and "L" is low. A blank is not applicable.

The specification shown here is based on facts and inferences derived from the generic system and any modifications which might have occurred in previous sessions (captured in the version tree of figure 4.3-11). The rankings reflect the amount of evidence gathered that a given factor is needed for a given function. The emphasis is on determining which factors are important to the system. Numerical goals, based on the metrics framework and comparisons to actual project data, will be established in later steps of the top-level roadmap.

ASQS

RANK FACTORS AND CRITERIA
 Assign Factor and Criteria Rankings

SOFTWARE QUALITY FACTOR IDENTIFICATION FORM INITIAL GOALS

<div style="transform: rotate(-45deg); display: inline-block;"> SOFTWARE QUALITY FACTOR SYSTEM OR SOFTWARE UNIQUE FUNCTION </div>	PERFORMANCE				DESIGN				ADAPTATION				
	EFFICIENCY	RELIABILITY	RECOVERABILITY	SURVIVABILITY	USABILITY	INTERFERENCE	MAINTAINABILITY	REPAIRABILITY	EQUIPABILITY	FLEXIBILITY	INTERDEPENDABILITY	PORTABILITY	REUSABILITY
STARTS SEE													
FINET APAB													
INFORMATION STORAGE													
INTEGRATION FRAMEWORK													
SYSTEM SOFTWARE													

How Continue

... Are software design characteristics that ease the verification of system operation following changes needed in order to satisfy the system mission. ... Maintenance of system software -- YES -- 00 CURRENT

... Are software design characteristics that ease the verification of system operation and performance needed in order to the system safety requirements for system software. UNKNOWN

... Are the software need to exhibit characteristics which provide for ... the quality and/or performance of system software in order to ...

... INTEROPERABILITY IS NEEDED -- the factor interoperability is needed for system software

PORTABILITY-IS-NEEDED-BS -- the factor portability is needed for system software

... 40) -- YES

PORTABILITY-IS-NEEDED -- the factor portability is needed for system software

FACTOR-IS-NEEDED-BS -- a factor is needed in system software

... 5) -- YES

FACTOR-IS-NEEDED-FM -- a factor is needed for system software

Figure 4.3-15. The User Zooms In To Review a Factor Ranking.

An important benefit of the Assistant is that it can "remind" users to specify factors which might otherwise have been inadvertently omitted from the specification. For example, from the mission area, the Assistant might infer that classified processing is involved, and in turn, that Integrity is important.

The user can review the reasoning used to conclude that a factor is needed. This allows users to build confidence in the specification as they review and refine the underlying reasons for the factor rankings. Here the user reviews the rationale for Maintainability of the Integration Framework by selecting a ranking menu item.

[i.e. HOW was it established that the factor maintainability is needed for integration framework?]

[1.1] RULE032 was used to conclude that the factor maintainability is needed for integration framework (0.25)

Thus, it has been established that the factor maintainability is needed for integration framework

ASQS													
Factor Rankings for START-EE													
RANK FACTORS AND CRITERIA													
Assign Factor and Criteria Rankings													
SOFTWARE QUALITY FACTOR IDENTIFICATION FORM - INITIAL GOALS													
SOFTWARE QUALITY FACTOR SYSTEM OR SOFTWARE UNIQUE FUNCTION	PERFORMANCE				DESIGN				ADAPTATION				
	EFFICIENCY	RELIABILITY	AVAILABILITY	SECURITY	MAINTAINABILITY	TESTABILITY	INTEROPERABILITY	PORTABILITY	ADAPTABILITY	EXTENSIBILITY	EVOLVABILITY	INTEGRATION	
FACTOR 1	L	M	M	M	L	M	M	L	M	M	M	M	M
FACTOR 2	L	M	M	M	L	M	M	L	M	M	M	M	M
FACTOR 3	L	M	M	M	L	M	M	L	M	M	M	M	M
FACTOR 4	L	M	M	M	L	M	M	L	M	M	M	M	M
FACTOR 5	L	M	M	M	L	M	M	L	M	M	M	M	M

FACTOR 1: INTEROPERABILITY IS NEEDED. The factor interoperability is needed for system software.

FACTOR 2: INTEROPERABILITY IS NEEDED. The factor interoperability is needed for system software.

FACTOR 3: PORTABILITY IS NEEDED. The factor portability is needed for system software.

FACTOR 4: PORTABILITY IS NEEDED. The factor portability is needed for system software.

FACTOR 5: PORTABILITY IS NEEDED. The factor portability is needed for system software.

FACTOR 1: INTEROPERABILITY IS NEEDED. The factor interoperability is needed for system software.

FACTOR 2: INTEROPERABILITY IS NEEDED. The factor interoperability is needed for system software.

FACTOR 3: PORTABILITY IS NEEDED. The factor portability is needed for system software.

FACTOR 4: PORTABILITY IS NEEDED. The factor portability is needed for system software.

FACTOR 5: PORTABILITY IS NEEDED. The factor portability is needed for system software.

Figure 4.3-16. Reviewing Suggestions by the Assistant Builds Confidence in the Specification.

Explanations for the factor rankings are displayed in the WHY window shown in the upper left hand corner (other windows will be described later in the scenario). Explanations allow the user to understand the rationale for a given ranking and refine the rankings by refining the underlying reasons.

Rankings are derived by the Assistant based on if-then rules that relate system characteristics and needs to software quality factors. The Assistant uses certainty factors to represent the confidence in conclusions. The certainty that a factor is needed is used as the basis for determining the factor rankings (H, M, L, and Not Applicable). In this instance rule 32 was used to conclude that Maintainability is needed for Integration Framework. Further explanation of how rule 32 was used can be obtained by selecting HOW, and indicating the item for further explanation (i.e., [1.1]).

EMC IN Configuration window: STAFF:EE

[i.e. HOW was RULE030 used?]

It has already been established that

- [3.1] the level of system availability (the portion of total operational time that the system performs or supports critical functions) needed for integration framework is one of high medium low and
- [3.2] software errors detected in integration framework must be quickly located and fixed in order to satisfy system availability requirements

Therefore

the factor maintainability is needed for integration framework is as follows

If the level of system availability (the portion of total operational time that the system performs or supports critical functions) needed for integration framework is:

- a) equal to high then: yes (0.50).
- b) equal to medium then: yes (0.35).
- c) equal to low then: yes (0.25).

ASQS

RANK FACTORS AND CRITERIA

Assign Factor and Criteria Rankings

QUALITY FACTOR IDENTIFICATION FORM INITIAL GOALS

SOFTWARE QUALITY FACTOR	PERFORMANCE					DESIGN					ADAPTATION				
	EFFICIENCY	RELIABILITY	AVAILABILITY	TRANSPORTABILITY	SAFETY	MAINTAINABILITY	INTEROPERABILITY	PORTABILITY	ADAPTABILITY	EXTENSIBILITY	EVOLVABILITY	MODIFIABILITY			
MAINTAINABILITY															

TYPE	RANK	ADAP	EFF	REL	AVAIL	TRANSP	SAFE	MAINT	INTEROP	PORT	ADAP	EXTEN	EVOLV	MODIF
MAINTAINABILITY														

EMC IN Configuration window: STAFF:EE

... (detailed reasoning chain for Rule 30) ...

EMC IN Configuration window: STAFF:EE

... (detailed reasoning chain for Rule 30) ...

Figure 4.3-17. Reviewing the Detailed Reasoning Chain Allows Users to Determine Refinements

Here the user further reviews the rationale for Maintainability. Rule 30 was used to relate system factor needs to the software factor Maintainability. System factors are system (hardware and software) quality terms such as Availability, Transportability, Safety, etc. If information is known about system factors, they can be used as one consideration for software quality factor needs.

The user can obtain further explanations for the items in the Why Window (e.g., [3.1], [3.2]). For example, further explanation of [3.2] would show that the user provided a response to a question. If the user disagrees with the response based on a greater understanding of the implications, the answer can be modified using the answer window (shown later).

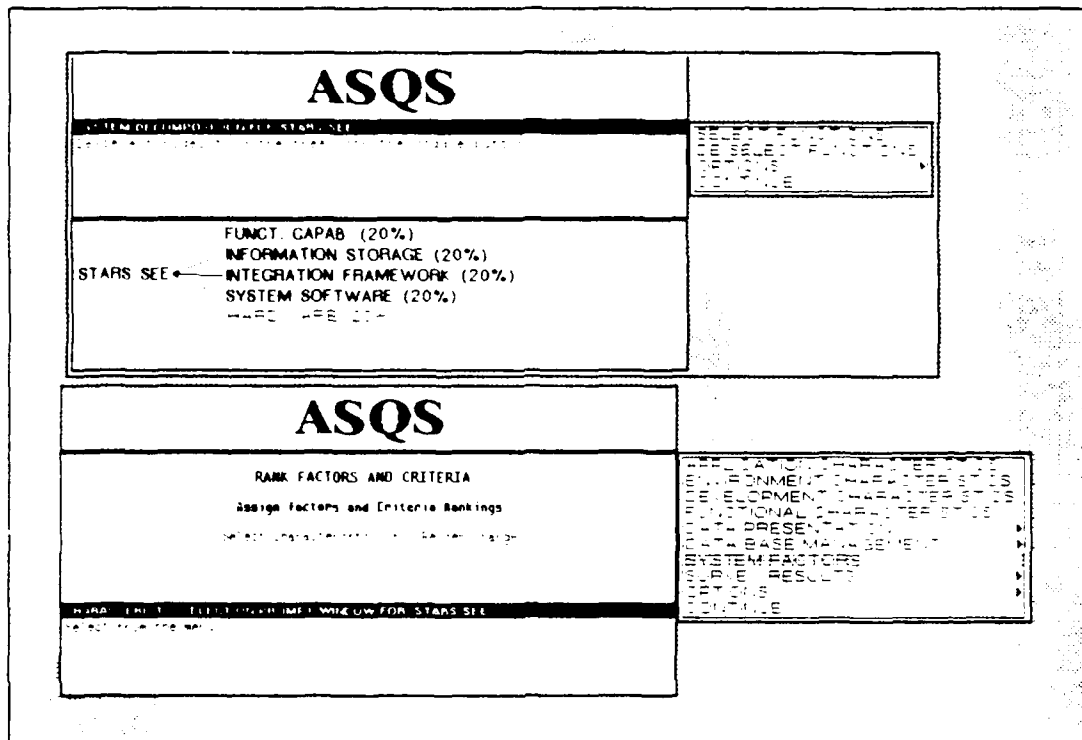


Figure 4.3-18. Quality Concerns May Be Selected from a Variety of Sources.

Now the user wants to review the answers in detail to refine the specification and make distinctions between individual functions. This state is obtained from the previous step by selecting CONTINUE, and REVIEW/CHANGE ANSWERS. A snapshot of the top window is shown together with the bottom window to illustrate the process of determining the functions to be reviewed and selecting the type of characteristics to be reviewed.

The characteristics are grouped into categories to allow the user to review specific aspects of the system. Characteristics provided in Section 4.1.2 of the Guidebook, Assign Quality Factors and Goals (Step 2), are supported by the Assistant. Other characteristics (e.g., Application Characteristics, Development Characteristics, Environment Characteristics, Functional Characteristics, etc.) are provided by DRC as a result of the Guidebook Validation effort. These characteristics are described in Section 4.1.1.2.

ASQS

RANK FACTORS AND CRITERIA
Assign Factors and Criteria Rankings
Select Characteristics for Review/Change

CHARACTERISTIC SELECTION PROMPT WINDOW FOR: START: SEE

ENVIRONMENT CHARACTERISTICS	SELECT
1. AIR QUALITY	<input type="checkbox"/>
2. WATER QUALITY	<input type="checkbox"/>
3. SOIL QUALITY	<input type="checkbox"/>
4. VEGETATION	<input type="checkbox"/>
5. WILDLIFE	<input type="checkbox"/>
6. CLIMATE	<input type="checkbox"/>
7. LAND USE	<input type="checkbox"/>
8. HISTORIC RESOURCES	<input type="checkbox"/>
9. CULTURAL RESOURCES	<input type="checkbox"/>
10. VISUAL QUALITY	<input type="checkbox"/>
11. SOUND	<input type="checkbox"/>
12. VIBRATION	<input type="checkbox"/>
13. AIR QUALITY	<input type="checkbox"/>
14. WATER QUALITY	<input type="checkbox"/>
15. SOIL QUALITY	<input type="checkbox"/>
16. VEGETATION	<input type="checkbox"/>
17. WILDLIFE	<input type="checkbox"/>
18. CLIMATE	<input type="checkbox"/>
19. LAND USE	<input type="checkbox"/>
20. HISTORIC RESOURCES	<input type="checkbox"/>
21. CULTURAL RESOURCES	<input type="checkbox"/>
22. VISUAL QUALITY	<input type="checkbox"/>
23. SOUND	<input type="checkbox"/>
24. VIBRATION	<input type="checkbox"/>

Figure 4.3-21. Lists of Concerns Help Ensure That All Issues are Considered by the User.

Environment characteristics are selected in the same manner as shown for application characteristics. Environment characteristics (including entries from Table 4.1.2-2 in the Guidebook) imply relevant factors as shown.

This state is obtained from the previous by selecting PREVIOUS STEP, ENVIRONMENT CHARACTERISTICS.

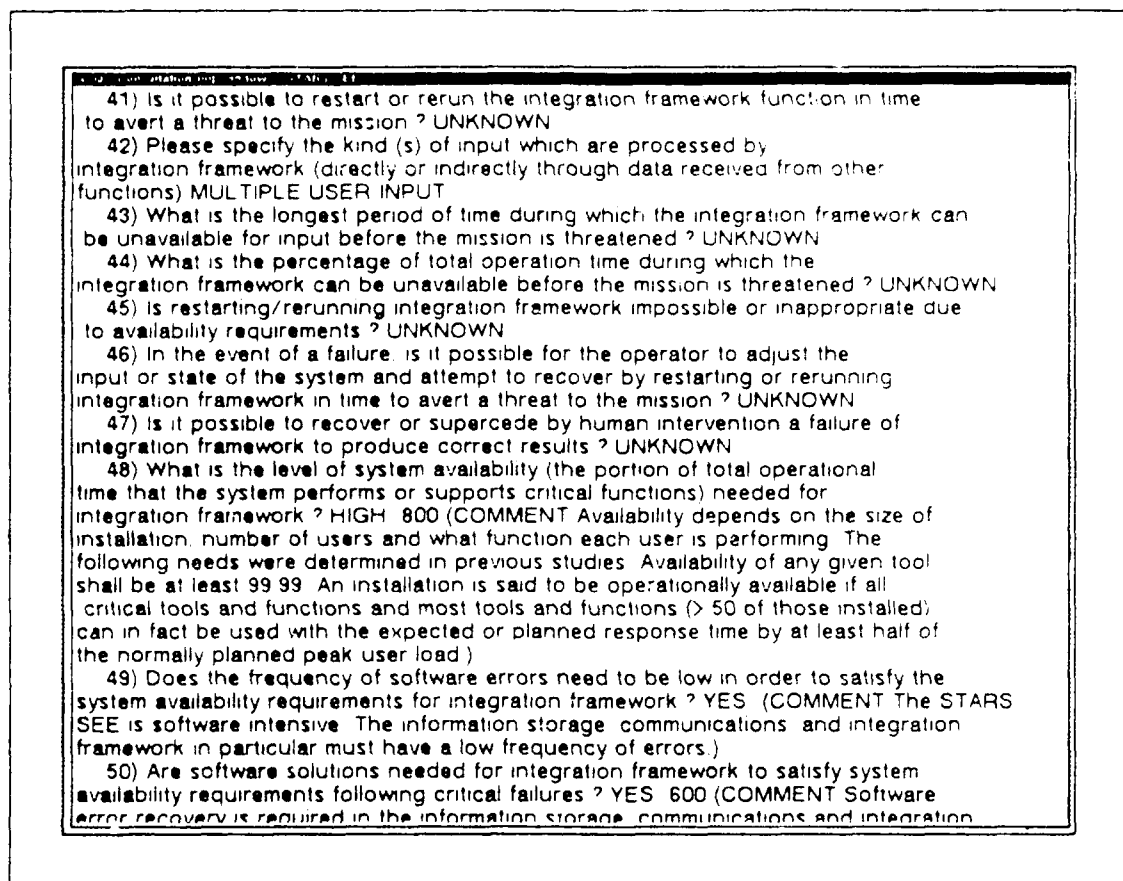


Figure 4.3-22. The Implications Of Changes To Answers Are Determined During A Consultation.

A consultation provides a dialogue between the user and the Assistant, which provides the basis for a common understanding of the system characteristics and needs and software quality concerns. The implications of changes are reviewed by running a consultation, which incorporates updated answers in the context of the overall knowledge of the system. This figure shows some of the kinds of questions that are asked. The answers, along with associated comments and certainty factors, are provided either through the answer window, or during the consultation itself.

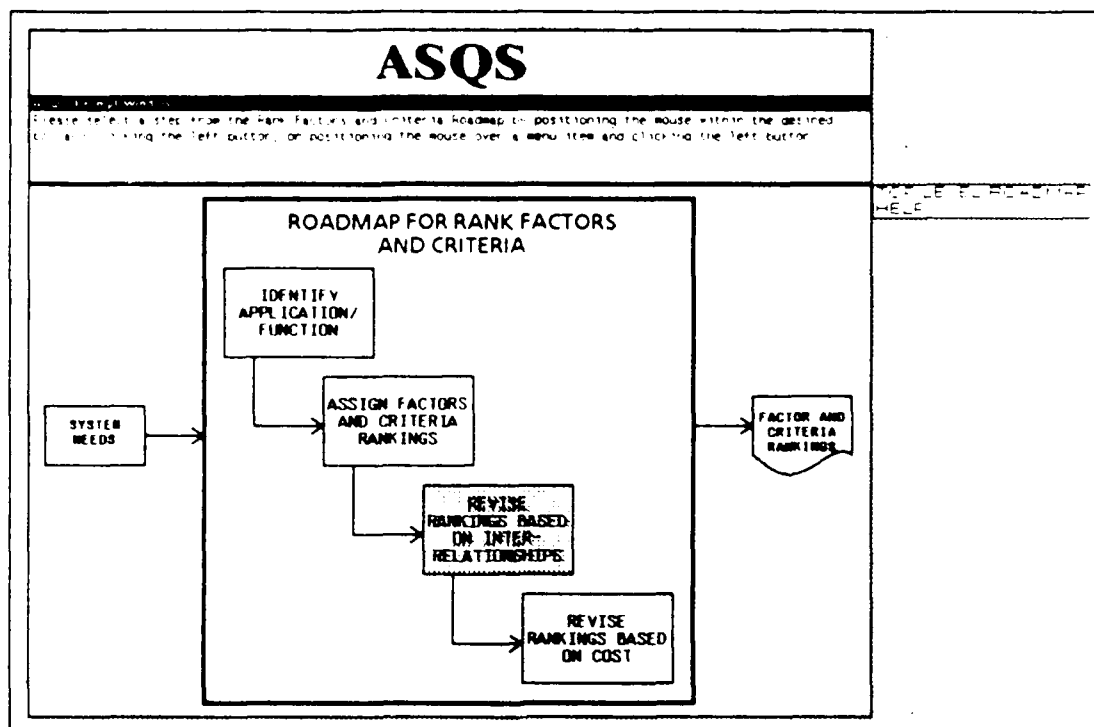


Figure 4.3-24. The Rankings are revised based on Factor Interrelationships.

The user can either work forward to revised rankings by browsing the interrelationships (using REVIEW/CHANGE ANSWERS) or work backwards by reviewing the revised rankings and isolating aspects and making changes (using CONSULT AND SHOW RANKINGS).

<h1>ASQS</h1>														
Factor Rankings for STARS SEE														
RANK FACTORS AND CRITERIA														How Continue
Assign Factor and Criteria Rankings														
SOFTWARE QUALITY FACTOR IDENTIFICATION FORM - INITIAL GOALS														
<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="transform: rotate(-45deg); white-space: nowrap;">SYSTEM OR SOFTWARE UNIQUE FUNCTION</div> <div>SOFTWARE QUALITY FACTOR</div> </div>		PERFORMANCE					DESIGN			ADAPTATION				
		EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	USABILITY	CORRECTNESS	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	INTEROPERABILITY	PORTABILITY	REUSABILITY
STARS SEE		L-3		H+4	+1	M	M+5	L	M	M	L	L		
FUNCTION CAPAB		L-3		H+4	+1	M	M+5	L	M	M	L	L		
INFORMATION STORAGE		L-3	M	H+4	L	M	M+5		M	M	L	M	L	
INTERACTION FRAMEWORK		L-3		H+4	L+1	M	M+5	L	M	M	L	M	L	
SYSTEM SOFTWARE		L-2		H+1	L	L	H+5		M	M		M	L	

Figure 4.3-25. Notes draw the user's attention to recommended changes based on Factor Interrelationships.

Notes in the form of "+n" or "-n" indicate the confidence that a given factor is recommended for overall consistency of the factor goals with respect to factor interrelationships. Factor interrelationships are documented in section 4.1.3. of the Guidebook.

The notes encourage development of consistent factor goals by highlighting the implications of factor interrelationships. A "-n" note indicates the confidence that achieving the factor will be difficult or impossible because of other factor goals. For example, efficiency is often in a negative relationship with maintainability. A "+n" note indicates the confidence that the factor is recommended because it contributes to the achievement of other factor goals. The integer number in each note represents the amount of confidence in the recommendation.

<h1 style="margin: 0;">ASQS</h1>													
1. Ranking for TOR: EE													
<h2 style="margin: 0;">RANK FACTORS AND CRITERIA</h2> <p style="margin: 5px 0;">Assign Factor and Criteria Rankings</p> <p style="margin: 0;">SOFTWARE QUALITY FACTOR IDENTIFICATION FORM - INITIAL GOALS</p>													
<div style="display: flex; flex-direction: column; align-items: center;"> <div>SOFTWARE QUALITY FACTOR</div> <div>SYSTEM OR SOFTWARE UNIQUE FUNCTION</div> </div>		PERFORMANCE			DESIGN			ADAPTATION					
		EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	USABILITY	CORRECTNESS	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	INTEROPERABILITY	PORTABILITY
STARS SEE		L-3		H+4	L+1	M	M+6	L	M	M	L	M	L

How
Continue

Figure 4.3-26. The user reviews reasons for the recommended change.

In this scenario the reason for recommending Correctness due to factor interrelationships is reviewed for the overall STARS SEE. This is accomplished by selecting the menu item "M+6".

ASQS

Factor Ranking for STARS SEE

RANK FACTORS AND CRITERIA

Assign Factor and Criteria Rankings

SOFTWARE QUALITY FACTOR IDENTIFICATION FORM INITIAL GOALS

<div style="display: flex; justify-content: space-between;"> <div style="transform: rotate(-45deg); transform-origin: center;">SOFTWARE QUALITY FACTOR</div> <div>SYSTEM OR SOFTWARE UNIQUE FUNCTION</div> </div>	PERFORMANCE					DESIGN		ADAPTATION				
	EFFICIENCY	RELIABILITY	AVAILABILITY	SECURITY	MAINTAINABILITY	ERROR HANDLING	EFFICIENCY	RELIABILITY	AVAILABILITY	SECURITY	MAINTAINABILITY	

STARS SEE

--	--	--	--	--	--	--	--	--	--	--	--	--

STARS SEE

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

STARS SEE

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

Figure 4.3-27. Each reason can be reviewed for Applicability.

Rule 66 was used to conclude that the factor correctness is needed for the STARS SEE. In a case such as this where a positive relationship applies the confidence that the factor is needed is increased. That is, there is evidence that the factor is needed because of positive interrelationships with other factors.

In the case of negative relationships, the confidence that the factor is needed is neither increased nor decreased. Difficulty in achieving a factor due to negative factor relationships does not necessarily decrease the needed for that factor. However, it does imply that the reasons for needing the factor should be carefully reviewed to determine if they are valid reasons.

The reasons for the note can be reviewed in greater detail using the HOW option as shown (HOW [1.1]).

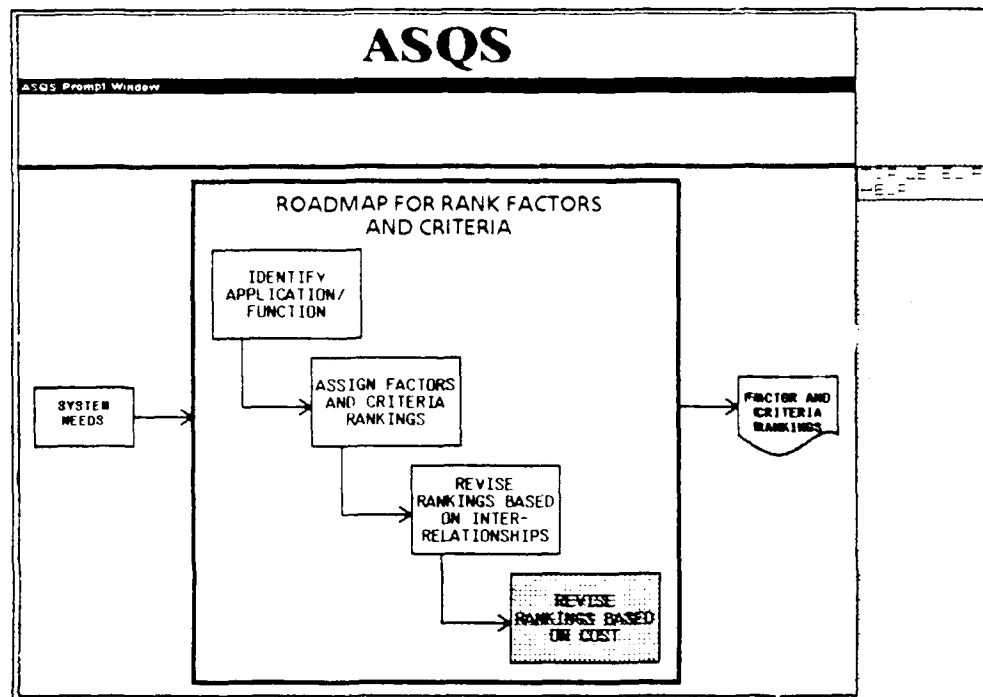


Figure 4.3-29. Revise Rankings based on costs.

The user can either work forwards towards revised rankings by browsing the recommendations based on costs, or work backwards by reviewing the revised rankings and isolating aspects and making changes.

This step considers the relative cost tradeoffs of building in a certain level of quality factor. Since there is very little available data on the cost of building in a given level of the factors, the analysis is relative. Notes in the factor ranking window highlight the factors which contribute most to the cost of achieving the overall goals. This analysis considers shared criteria and makes the simplifying assumption until further data is available that the cost of building in quality is the same for each of the 26 criteria.

This state is obtained from the previous state shown in the figure by selecting CONTINUE, REVISE RANKINGS BASED ON COSTS.

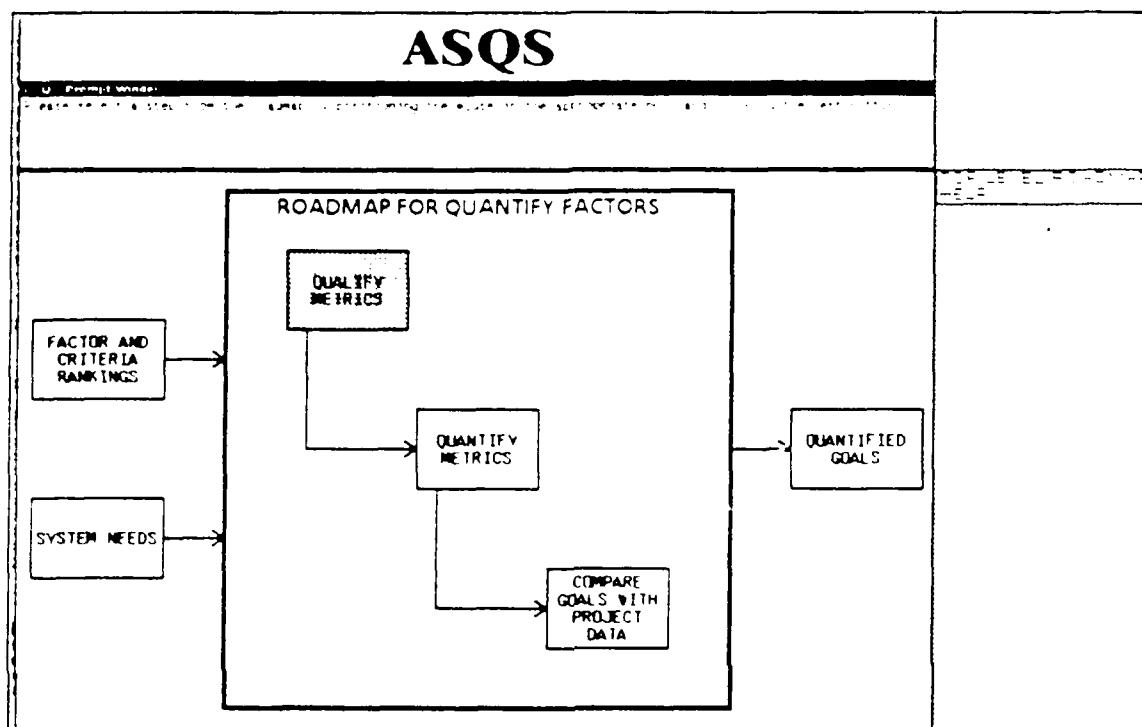


Figure 4.3-30. A Second-Level Roadmap illustrates the steps of Quantify Factors.

Quantify Factors consist of Qualify Metrics, Quantify Metrics, and Compare Goals with Project Data. Qualify Metrics involves selecting and weighting the applicable metrics and metric-elements (Quality Metrics -- Section 4.3 of the Guidebook). This process is often referred to as tailoring the framework. Quantify Metrics involves determining expected scores to develop quantified Factor goals in terms of the framework. Finally, historical project data is reviewed with respect to the quantified goals to determine if the factor goals are likely to achieve the desired results in actual quality rates. Qualify and Quantify Metrics can be revisited to revise the factor goals based on comparisons with project data. An overview of the steps of Quantify Factors is provided in Section 4.1.2.

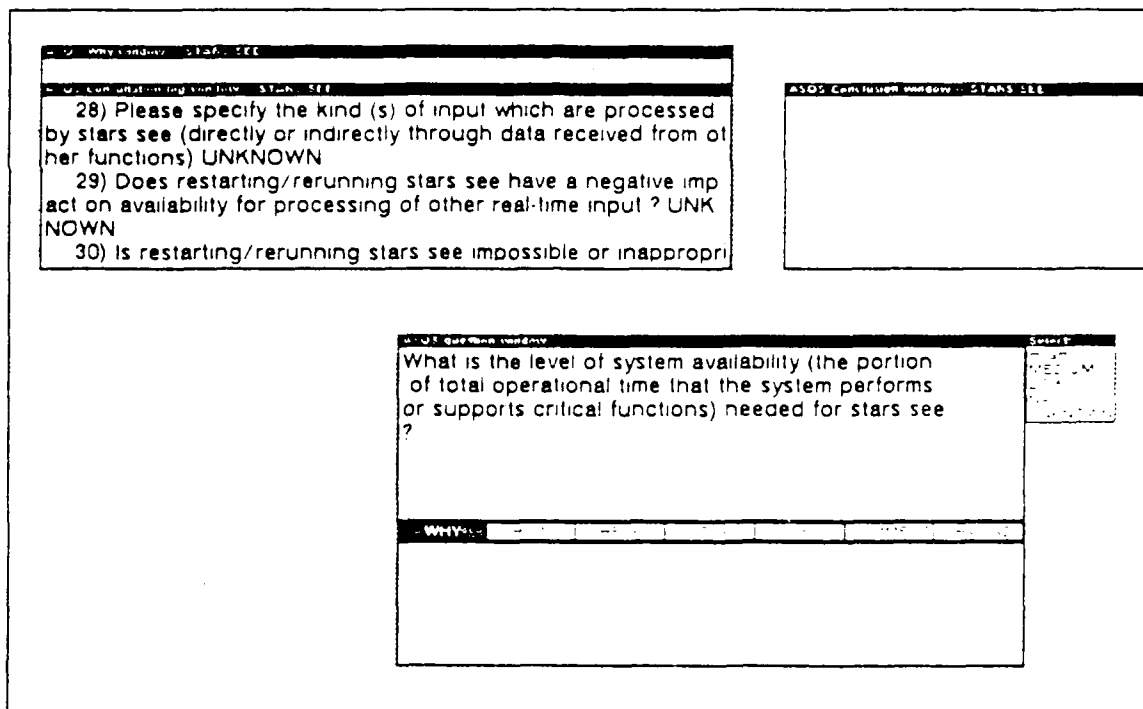


Figure 4.3-32. The user can ask the Assistant for explanations of the questions.

The process of answering acquisition questions is shown. This question, although not specifically related to a metric, might come up during Qualify Metrics if was not answered previously and it is in the chain of reasoning to determine if a metric is applicable (for example, a Reliability metric). Before the questions are answered, the applicability of individual metrics is inferred based on the information gathered so far, with a confidence level based on the certainty of that information and the inferences. The user answers questions to increase the certainty of the applicability.

In this example, the user asks why the question is asked. The Assistant answers WHY to help the user understand the purpose of the questions.

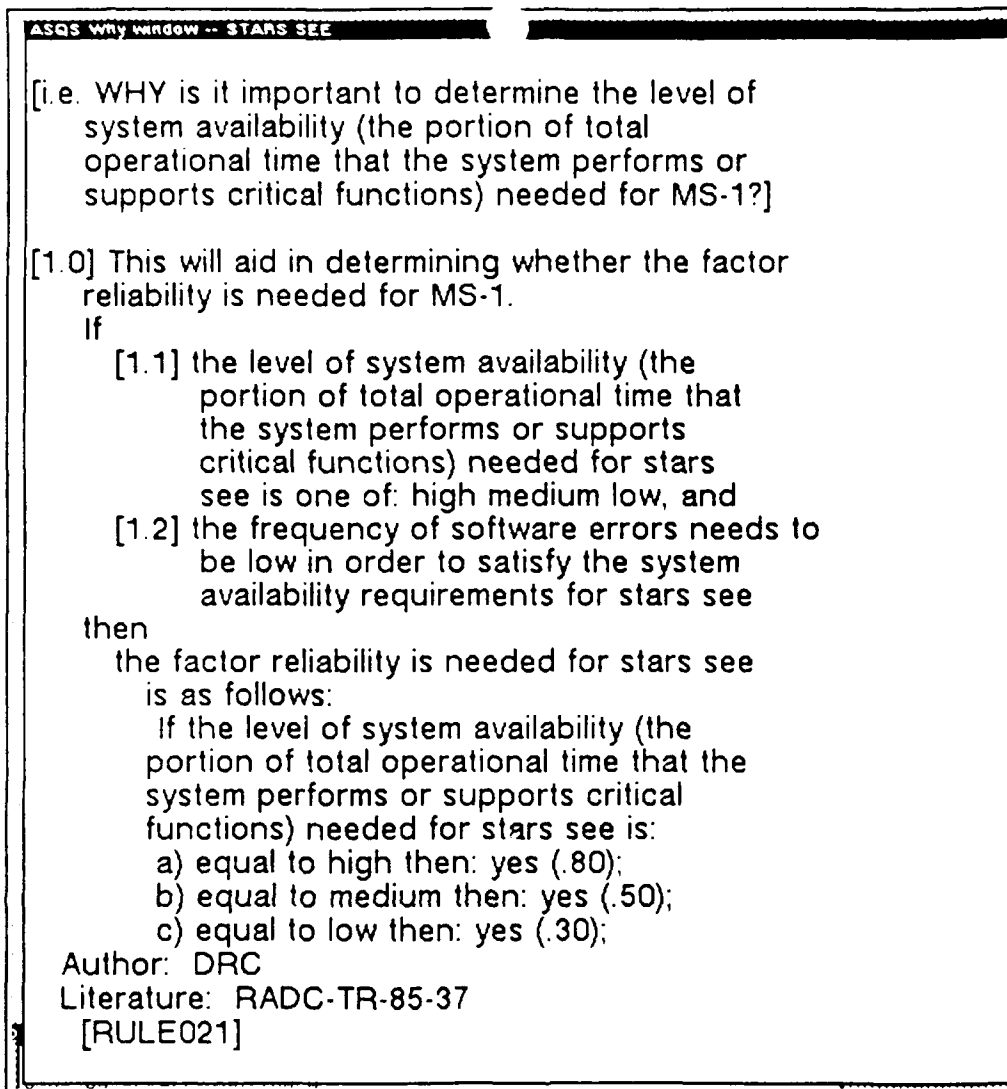


Figure 4.3-33. A WHY Explanation is Shown.

Explanations for the questions are presented in terms of the current goal of the consultation. For example, the current question is asked in order to determine if the factor Reliability is needed. This in turn might aid in determining the applicability of certain broad scope Reliability metrics that are known to apply whenever Reliability is needed.

HOW can be used to determine how the premises were or will be concluded.

<h1>ASQS</h1>													
RANK FACTORS AND CRITERIA													
Assign Factor and Criteria Rankings													
SOFTWARE QUALITY FACTOR IDENTIFICATION FORM - INITIAL GOALS													
<div style="display: flex; justify-content: space-between;"> <div style="transform: rotate(-45deg);">SYSTEM OR SOFTWARE UNIQUE FUNCTION</div> <div>SOFTWARE QUALITY FACTOR</div> </div>	PERFORMANCE				DESIGN				ADAPTATION				
	EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	USABILITY	CORRECTNESS	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	INTEROPERABILITY	PORTABILITY	REUSABILITY
STARS SEE	L-3		H+4	+1	M	M+6	L	M	M	L	L		
FUNCY CAPAB	L-3		H+4	+1	M	M+6	L	M	M	L	L		
INFORMATION STORAGE	L-3	M	H+4	L	M	M+5		M	M	L	M	L	
INTEGRATION FRAMEWORK	L-3		H+4	L+1	M	M+6	L	M	M	L	M	L	
SYSTEM SOFTWARE	L-2		H+1	L	L	H+5		M	M		M	L	

Show Tailored Framework
 How
 Continue

Figure 4.3-34. The Tailored Framework can be reviewed.

The tailoring resulting from the acquisition questions can be reviewed directly by the user. This is accomplished by selecting a menu item indicating the factor and function to be reviewed. In this scenario the menu item "H" for Reliability of Information Storage is selected.

ASQS	
QUANTIFY FACTORS Qualify Metrics Select Phase	SYSTEM/SOFTWARE REQUIREMENTS ANALYSIS PRELIMINARY DESIGN DETAILED DESIGN CODING AND UNIT TESTING INTEGRATION AND TESTING LEVEL TESTING SYSTEM INTEGRATION AND TESTING SYSTEM PERFORMANCE TESTING DECOMMISSION
SELECT PHASE WHEN READY FOR START: SEE THE PHASES OF THE LIFE CYCLE OF THE SYSTEM IN THE ASQS USER GUIDE	

Figure 4.3-35. The Tailored Framework is Reviewed By Phase.

The select phase window allows the user to review the tailored framework by individual life-cycle phase. This is necessary because the underlying metrics framework varies by phase, from System/Software Requirements Analysis to System Performance Testing.

The user can conveniently review the framework for different phases by using the option PREVIOUS STEP to return to the select phase window after the framework is reviewed.

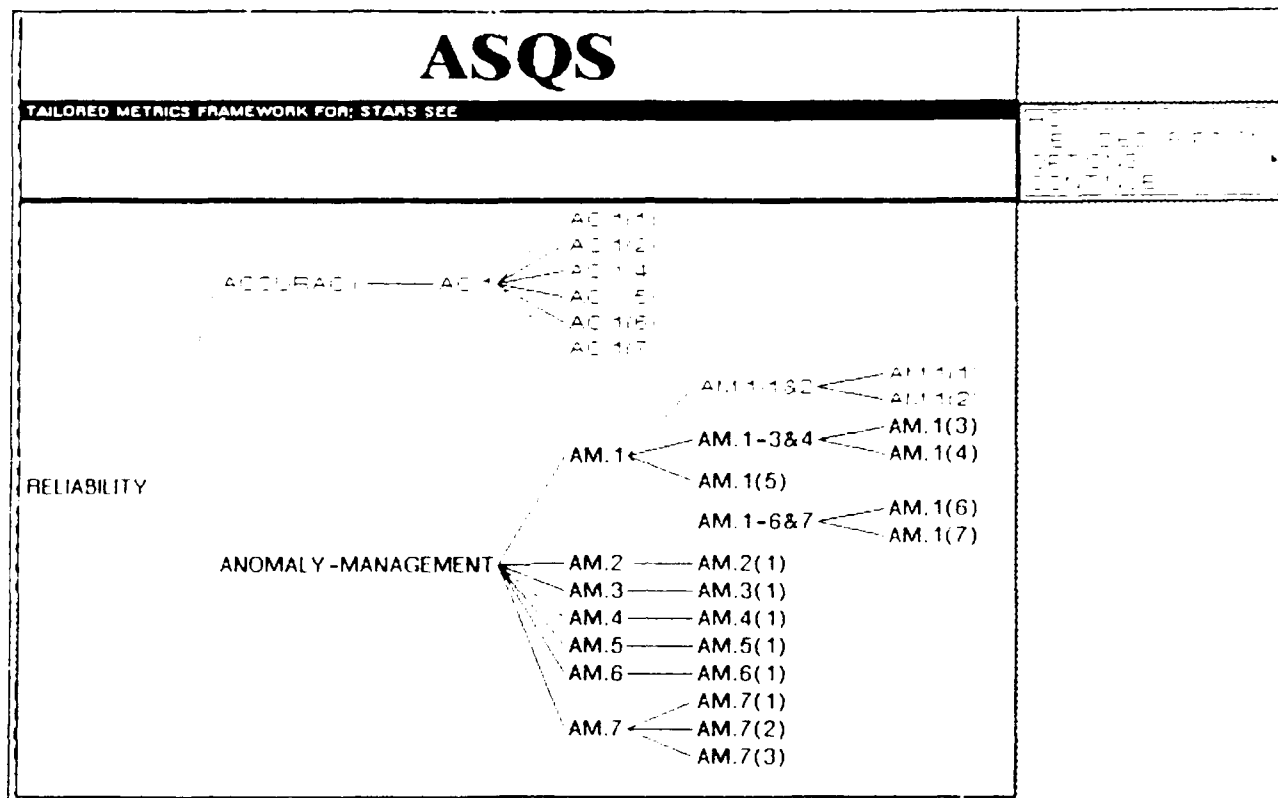


Figure 4.3-36. The Tailored Framework is shown in hierarchical form.

The metrics framework for the selected factor is shown, with factors, criteria, metrics and data items in bold when they are applicable to the selected function (i.e., they are tailored in). Here Anomaly Management metrics are applicable except AM.1-1&2 and its associated data items. AM.1-1&2 is not applicable because concurrent processing is not involved in the selected function.

As with other tree windows the user can move the tree around in the window by selecting a node which is to be centered in the window. The user can determine how a tree element was determined to be applicable during the consultation by using the HOW menu item. A description of each tree element is available using the VIEW DESCRIPTION option.

ASQS		DESCRIPTION
<p>RELIABILITY 17(05) 85(4) ANOMALY-MANAGEMENT 17(06) 85(4)</p> <p>AM 1 2(4) 8(4) AM 1(5) AM 2(2) AM 2(3) AM 2 2(4) 8(4) AM 2(4) AM 2(5) AM 2(6) AM 3 25(4) 10(4) AM 3(1) 00(4) 10(4) AM 3(2) 00(4) 10(4) AM 3(3) 10(8) 10(8) AM 3(4) 00(4) 10(4) AM 4 00(1) 10(4) AM 5 2(4) 8(4) AM 6 2(4) 8(4) AM 7 2(06) 8(4)</p>		

Figure 4.3-37. Numerical Factor Goals are provided automatically based on the Framework and user interaction.

The Assistant uses the available information to score the tailored framework and produce the table shown here. Scoring lends credibility to the specified numerical goals and decreases the likelihood that discrepancies between specified and evaluated scores are caused by goals that are too high.

The expected best and worst case scores are shown after each tree element with the associated confidence factor in parentheses. The user answers acquisition questions to build confidence in the numerical goals and narrow the range of expected scores. The questions are similar to those shown for Qualify Metrics. The user can review scores by life-cycle phase.

This state is obtained from the top-level roadmap by selecting QUANTIFY FACTORS, QUANTIFY METRICS, CONSULT AND SHOW RANKINGS, SHOW QUANTIFIED FRAMEWORK, and DETAILED DESIGN.

4.4 Computer System Characteristics

The Assistant is composed of both hardware and software components described in the following sections.

4.4.1 Hardware

The Assistant is a single desk-top workstation composed of the following hardware (Also see Figure 4.4-1):

- o A high resolution display with graphics capability
- o A keyboard and mouse for data entry
- o A printer capable of printing the screen in any state
- o A floppy disk drive
- o A hard disk drive.

The hardware selected for implementation of the Assistant shall have spare disk and CPU capacity to allow future expansion of the knowledge base as additional project information becomes available.

4.4.2 Software

In order for the Assistant to perform all functions allocated to the computer equipment the following software is required:

- o A vendor supplied environment supporting:
 - Pop-up menus
 - Window manipulation (scrolling, moving, sizing, and overlaying)
 - Control of all hardware components such as the disk and display screen, and printer
 - File management functions (read, write, open close, save, etc.).
- o A lisp compiler and interpreter to support knowledge-based representation and reasoning.

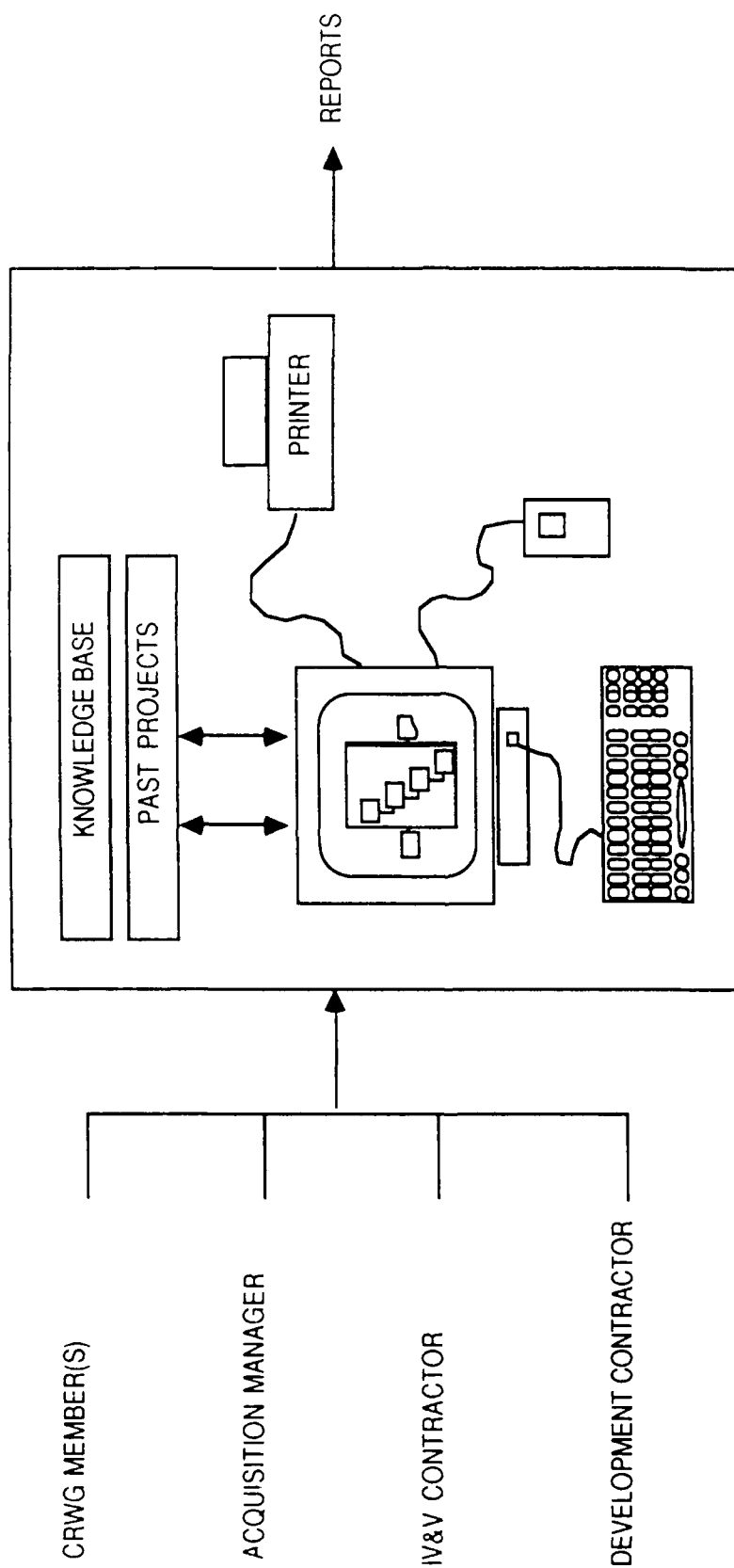


FIGURE 4.4-1 SYSTEM CONFIGURATION

5. GLOSSARY

The following glossary items have been derived from selected documents referenced in the Referenced Documents Section contained in this specification. Certain glossary items have been adapted to meet the needs of the Assistant.

Actual Parameter: An actual parameter is the particular entity associated with the corresponding formal parameter by a subprogram call, entry call, or generic instantiation. The association of actual parameters with formal parameters can be specified by named associations, by positional associations, or by a combination of these.

ADL: Ada Design Language.

Argument List: A list of data elements that specify the input and output parameters used during execution of a software unit.

Body: A body defines the execution of a subprogram, package, or task. A body stub is a form of body that indicates that this execution is defined in a separately compiled subunit.

Compilation Unit: A compilation unit is the declaration or the body of a program unit, presented for compilation as an independent text. It is optionally preceded by a context clause, naming other compilation units upon which it depends by means of one or more "with" clauses.

Computer Software Component (CSC): A functional or logically distinct part of a Computer Software Configuration Item (CSCI). Computer Software Components may be top-level, or lower-level.

Computer Software Configuration Item (CSCI): A part of a system, segment, or prime item. A CSCI is a Configuration Item (CI) which consists of one or more Top Level Computer Software Components (TLCSCs). CSCI(s) shall form the third level of the DOD-STD-2167A logical software hierarchy. Also see Configuration Item.

GLOSSARY (continued)

Configuration Item (CI): (1) A collection of hardware or software elements treated as a unit for the purpose of configuration management. (2) Hardware or software, or an aggregate of both, which is designated by the contracting agency for configuration management (DOD-STD-2167A).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Format: The positioning, packing, or organization of the order in which the data appears.

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

High Order Language: A programming language that 1) usually includes features such as nested expressions, user-defined data types, and parameter passing not normally found in lower order languages, 2) does not reflect the structure of any one given computer or class of computers, and 3) can be used to write machine-independent source programs. A single, higher-order language statement may represent multiple machine operations.

Interface: A connection or common boundary between two systems, devices, or elements of a single system. The connection may be used to complete an operation, expand the capabilities, or acquire information from one device or program to another. Two software units have an interface if they share information (declaration or data).

Lines of Code: The number of lines of source code, excluding comment lines and blank lines.

Local Variable: A variable which is declared in the main declaration section of a program unit and may only be referenced within the program unit.

GLOSSARY (continued)

Lower-Level Computer Software Component (LLCSC): Shall consist of a logical grouping of other LLCSCs or Units. LLCSCs shall form the second-lowest level of the DOD-STD-2167A logical structure.

Mission-Critical Function: A feature essential to fulfilling the desired objectives of the system.

Module: A discrete unit of source code that can be compiled independently. A module is designed to perform a defined function and has a defined set of interfaces with any other portions of the program. Modules in Ada are 1) subprogram specification and body, 2) package specification and body, 3) generic instrumentation, 4) task specification and body, and 5) generic specification and body.

Network: A system of computers, terminals, and data bases that are linked/interconnected with the use of communication lines.

Object: An object contains a value. A program creates an object either by elaborating an object declaration or by evaluating an allocation. The declaration or allocator specifies a type for the object: the object can only contain values of that type.

Package: A package specifies a group of logically related entities, such as types, objects of those types, and subprograms with parameters of those types. It is written as both a package declaration and a package body. The package declaration has a visible part, containing the declarations of all entities that can be explicitly used outside the package. It may also have a private part containing structural details that complete the specification of the visible entities, but which are irrelevant to the user of the package. The package body contains implementations of subprograms (and possibly tasks and other packages) that have been specified in the package declaration. A package is one of the kinds of program unit.

GLOSSARY (continued)

Parameter: A parameter is one of the named entities associated with a subprogram, entry, or generic unit, and used to communicate with the corresponding subprogram body, accept statement, or generic body.

Project: A planned undertaking of something to be accomplished, produced, or constructed, having a finite beginning and a finite ending.

Software Development and Maintenance Environment (SDME): An integrated set of automated tools, computing machinery, guidance documents, and people that collectively support the creation and evolution of WIS software throughout its life cycle. The object of the SDME is to increase productivity and to improve software quality throughout the life cycle of WIS software.

Software Development Plan (SDP): This plan defines the software development approach, the management guidelines to be followed on the software project, the division of responsibilities between organizations working on the project, the technical status reporting responsibilities and formats, and other management practices.

Software Life Cycle: The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirements phase, design phase, implementation phase, operational and maintenance phase, and sometimes, retirement phase.

Software Trouble Report (STR): A report which is initiated when a problem is encountered with a software product. The STR describes the symptoms/cause of the problem, the execution environment, and the problem originator. The STR is also used to document the problem resolution, action taken, and the individual(s) developing and verifying the fix.

Spare Memory: Main memory allocated for a program and which is not being used to store and process the program or its data.

GLOSSARY (continued)

Subprogram: A subprogram is either a procedure or a function. A procedure specifies a sequence of actions and is invoked by a procedure call statement. A function specifies a sequence of actions and also returns a value called the result, so a function call is an expression. A subprogram is written as a subprogram declaration, which specifies its name, formal parameters, and (for a function) its result; and a subprogram body which specifies the sequence of actions. The subprogram call specifies the actual parameters that are to be associated with the formal parameters. A subprogram is one of the kinds of program unit.

Subsystem: A self-contained portion of a system that performs one of the major system functions, usually with only minimal interaction with other portions of the system.

Synchronization: The process of ensuring that two or more components of a system are ready and capable of communicating with one another.

System: A group of units (hardware/software) combined to form a whole and to operate in unison to perform a desired function. The first level of the DOD-STD-2167A logical software hierarchy.

Test: The execution of documented procedures to verify a functional, performance, or human factors characteristic of a specified program or set of programs.

Top Level Computer Software Component (TLCSC): Shall consist of a logical grouping of Lower-Level Computer Software Components (LLCSC) or Units. TLCSC(s) shall form the fourth level of the DOD-STD-2167A logical software hierarchy.

Type: A type characterizes both a set of values, and a set of operations applicable to those values. A type definition is a language construct that defines a type. A particular type is either an access type, an array type, a private type, a record type, a scalar type, or a task type.

GLOSSARY (continued)

Unit: The lowest level of the DOD-STD-2167A logical hierarchy structure specified in the Detailed Design and which completely describes a function in sufficient detail to allow implementing code to be produced and tested independently of other Units. Units are the actual physical entities implemented in code.

Unit Test: This software development phase includes preparation and execution of unit test cases to verify that the individual code units function according to their design. Also see Unit.

WIS: Worldwide Military Command and Control System, Information System.

APPENDIX A

Sample Rules for Tailoring Portability

The following rules represent a subset of the rules that can be used to prove the applicability or non-applicability of criteria, metrics and metric-elements comprising the factor Portability. The Confidence factors are shown in parentheses following the "then".

```
If
  Need-for-Independence-P /* Criteria */
Then (1.0)
  Need-for-Portability /* Factor */

if Plan-to-use-hardware-with-varying-word-size
then (1.0)
  Need-machine-independence /* Metric */

if Plan-to-use-different-compiler(C)
  and Compiler-is-really-different(C)
then (1.0)
  Need-system-software-independence /* Metric */

if standard-subset-is-required /* Metric-element */
then (0.6)
  not Compiler-is-really-different

if standard-subset-is-required
  and standard-subset-is-defined /* Metric-element */
then (0.8)
  not Compiler-is-really-different

if Mission-critical-application
then (0.5)
  Ada-is-required

if Ada-is-required
then (1.0)
  standard-subset-is-required
  standard-subset-is-defined

if Plan-to-use-different-run-time-services
then (1.0)
  Need-system-software-independence

if Plan-to-use-different-libraries
then (1.0)
  Need-system-software-independence
```

```

if Plan-to-use-different-hardware
then (0.5)
    Plan-to-use-hardware-with-varying-word-size

if Plan-to-use-different-hardware
then (0.6)
    Plan-to-use-different-compiler

if Plan-to-use-different-hardware
then (0.8)
    Plan-to-use-different-run-time-services

if
    Need-machine-independence
then (1.0)
    Need-independence

if
    Need-system-software-independence
then (1.0)
    Need-independence

if Plan-to-use-hardware-with-varying-word-size
    and word-size-dependencies
then (1.0)
    Need-to-port-machine-dependencies

if
    Need-to-port-machine-dependencies
then (1.0)
    Need-to-provide-dependent-services-in-software-layer
    or Need-to-modify-code-to-run-on-different-hardware

if
    Need-to-provide-dependent-services-in-software-layer
then (1.0)
    Need-to-determine-functionality-of-dependent-services

if Need-to-determine-functionality-of-dependent-services
then (1.0)
    Need-dependent-services-documented-separately
    or Need-to-locate-dependencies-and-determine-their-use

if Need-to-locate-dependencies-and-determine-their-use
then (0.7)
    Need-comments-to-indicate-use-of-dependencies
    and Need-comments-to-explain-use-of-dependencies

if Need-to-locate-dependencies-and-determine-their-use
    and Dependencies-can-be-isolated-to-some-extent
then (0.8)
    Need-Modularity

```

```
if Need-comments-to-indicate-use-of-dependencies
then (1.0)
    Need-self-descriptiveness

if Need-comments-to-explain-use-of-dependencies
then (1.0)
    Need-self-descriptiveness

if Need-to-modify-code-to-run-on-different-hardware
and Dependencies-can-be-isolated-to-some-extent
then (0.9)
    Need-Modularity
```